ALGORITHMS FOR POST-SILICON VALIDATION AND DEBUG OF RADIO-FREQUENCY, ANALOG, AND MIXED-SIGNAL CIRCUITS AND SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Barry John Muldrey

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

August 2019

ALGORITHMS FOR POST-SILICON VALIDATION AND DEBUG OF RADIO-FREQUENCY, ANALOG, AND MIXED-SIGNAL CIRCUITS AND SYSTEMS

Approved by:

Dr. Abhijit Chatterjee
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Jennifer Hasler
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Hua Wang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Gregory Durgin
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Adit Singh
School of Electrical and Computer Engineering
Auburn University

Date Approved: April 29, 2019

Every artist, every scientist, every writer must decide now where he stands. The artist must take sides. He must elect to fight for freedom or for slavery. I have made my choice. I had no alternative.

Paul L. Robeson

I would like to dedidate this thesis to the legacy of my mother, Cynthia Gail Newman. Though she left this earth quite a bit early, she also left upon me the understanding that I should take time to experience a bit of joy in spite of whatever other unpleasantness life might present.

ACKNOWLEDGEMENTS

I would like to acknowledge:

My wife, Ashley King, for reminding me every day that life extends beyond the bounds of mathematics, statistics, and the like. She's been a most exciting partner for the last few years, and I look forward to the remaining one hundred and fifty or so.

My father, Barrie Muldrey, for trying his hardest to teach me commitment. I would question whether any of it sunk in, but this thesis might suggest some did.

My advisor, Dr. Abhijit Chatterjee, and all the wonderful colleages of KL-1359: Dr. Sabyasachi Deyati, Dr. Suvadeep Banerjee, Dr. Debesh Bhatta, Dr. Joshua Wells, Dr. Jayaram Natarajan, and Dr. Nicholas Tzou among others.

Dr. Jennifer Hasler for her continual support of my work and her keen perspective on the academic process.

My colleague, Dr. Michael Giardino, with whom I share a great deal and from whom I've learned an innumerable quantity. Dr. Giardino and his wife Rachel recently gave birth to their son, Alessandro, whom I'd like to commemorate here.

The support and company of many friends, a collection of brilliant individuals in whose number I most surely don't belong: Matthew Elias, Matthew Resignola, Sean Munro, Eric Leefe, Neal Stastny, Dr. Sarp Satir, Neal Stastny, Dr. Hommood Al Rowais, Dr. Joshua Fulwiler, James Sticker III, Dr. David Torrello, Karl Peterson, Dr. Christopher Phaneouf, and Dr. Nortey Yeboah among others.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Abstraction has provided us a mechanism by which to create systems of astounding complexity. Digital system design and validation tools have leveraged abstraction to the degree that one single algebra can be used to model everything from a single gate to a cloud server. We can in theory reduce any digital component to a single analytical expression. Further, we have a good understanding of the limits of that abstraction and models for those phenomena that challenge its validity: timing margins, drive strength, and clock jitter for example.

Advances in the fields of digital testing and design verification have produced tools which are capable of generating sets of digital patterns which can test an arbitrary block of digital functionality with a quantifiable coverage. And so, once the fundamental validity of the analytical algebraic model has been established (or assumed), one can reliably build confidence in a designs' likelihood of success prior to first silicon.

The field of Analog, RF, and mixed-signal design (analog and mixed-signal (AMS)) doesn't enjoy the unified abstraction paradigm that digital circuits do, and so it lacks a comprehensive approach to design validation, testing, and debugging. Generally, circuit and component designers will test the performance of their circuits under hand-crafted sets of conditions and process corners to ensure that performance specifications maintained. They will then provide another bespoke, fixed, lightweight numerical model of their system for downstream integration simulations. Recently, this strategy is proving insufficient. Other proposed methods address various aspects of the problem without a cogent, quantifiable analytical framework around the entirety of the problem.

The work presented in this thesis makes inroads toward a unified paradigm for the verification of hierarchical dynamical models and provides various numerical and statistical algorithms for the abstraction of AMS circuits, for quantifying dynamics "lost" in the process of abstracting, for the provision of test vectors for composite AMS systems, and for enabling a unified fault localization scheme.

The primary motivation is demand for techniques which leverage available system models without requiring either absolute accuracy or completeness in order to generate improved system-level tests. Methodologies are provided which benefit pass/fail production testing as well as pre- and post-silicon debugging and diagnostic testing of AMS systems.

INTRODUCTION

Advancements in semiconductor manufacturing have enabled the production of transistors so small that they cease to behave deterministically; they have also enabled the integration of increasingly diverse kinds of systems onto smaller and smaller pieces of silicon. These technological developments have been the primary mechanism through which greater electronic system performance and lower system cost have been achieved. Simultaneously, design tools have progressed in parallel with manufacturing techniques, focused on managing increasing transistor counts. Tools were developed to allow relatively small teams of perhaps hundreds of engineers design and manipulate the interactions of billions of transistor devices. Testing methodologies were developed so that the designers could quantify their confidence in their output at various stages. Historically, testing techniques have centered around direct measurement of performance metrics.

Until recently, the trajectories of process, design, and testing tools have consistently paid dividends. Now, component-level designs are failing to integrate, system-level performance suffers from unforeseen component interactions, and fabricated chips exhibit behaviors that no model has predicted. Systems spend more time in debugging and diagnosis phases, and preproduction engineering costs are increasing. After reaping the low-hanging fruit of planar transistor scaling, designers of both electronic systems and design tools face new sets of challenges [1].

There are three complementary factors contributing to the frustration of existing approaches. For one, shrinking device sizes and shrinking power budgets leave components more sensitive to interactions with neighboring components. Interactions between components are difficult to model because they are often a function of downstream decisions like chip layout, and because they require simultaneous simulation of all interacting components. For another, extreme performance expectations require operation within increas-

ingly narrow performance tolerances (i.e. higher transmission rates generally require lower noise). This means that all subsystems must be increasingly robust against smaller and smaller mechanisms which might disrupt their performance – disruptions which are difficult to predict and model. Third is the simulation complexity problem. In order to simulate larger numbers of interacting components in reasonable time, the computational complexity of individual component models must be reduced. High-order behavioral phenomena should be abstracted away to expedite evaluation; however, it is some of these "high-order effects" which are leading to downstream integration problems; they are increasingly relevant to overall system performance.

Because compact analytical expressions are not readily available for low-level models, practicable solutions must operate with a basis in the discrete empirical observational data resulting from simulation or otherwise acquired through experimentation. It should be emphasized though, that the observational data are reflective of both the properties of the system under analysis *and properties of the test stimulus*, and so a mechanism for distinguishing between inferred system properties and properties of particular stimuli is required. A complete suite of tools should provide guidance for simulation setup or experimental design and test stimulus selection as sell as metrics for measuring the quality or utility of the resulting data. Such a measurement implicitly provides a measure of quality for the test itself (setup and stimulus).

The fundamental contributions of this work are:

- The synthesis of test stimulus generation and incremental model building applied to the system validation problem on the dynamical level.
- A holistic approach toward fault localization which does not require invertiblity of system components.
- The demonstration of the aforementioned methodologies in both pre- and post-silicon validation contexts.

The organization of this thesis in intended to follow the chronological arc of my research. It begins with a very acute focus on test stimulus design, in the vein of the "alternate testing" of my forebears, then explores reinforcement learning as a mechanism for capturing and utilizing the "data exhaust" of iterative optimization, and concludes with a more holistic view of test data and *data quality* in a chapter exploring the application of a design-of-experiments approach to test design.

Chapter 1 presents a summary of the state-of-the-art in hierarchical circuit design and modeling, design validation, and diagnostic techniques as they apply to AMS systems. It discusses in depth the factors that threaten the continued success of status-quo approaches to AMS design and validation and critiques several proposed trajectories of work. Chapter 2 discusses several optimization-based resolutions to some of the contemporary challenges. Chapter 3 presents an investigation of ways contemporary artificial intelligence tools might be of use. Chapter 4 presents an alternative formulation of some of the challenges facing AMS validation as a "design of experiments" problem, and applies some statistical tools from that domain. Chapter 5 outlines the extension of validation techniques into fault localization and debugging applications. Chapter C details several substantial software products which have been developed and opensourced for use by both AMS practitioners and the greater scientific community.

CHAPTER 1
KEY CONCEPTS

The most critical component of any study is establishing the foundational premises by formalizing the problem; in doing so one makes explicit the degrees-of-freedom of the problem and identifies measures of success. This formalization process proved to be one of the most difficult aspects of my work. There have been several efforts in the past toward formalizing the problem of AMS validation all of which fail to either capture all features of the problem, or reduce them away to such point that they're no longer meaningful in relation to the problems faced in contemporary hierarchical circuit design [2], [3], [4].

First, we'll define some terms and look at some basic properties of test stimuli along with some examples, then we'll look at a formal definition of "validation" provided by the authors of [4] and use it to frame the work presented in the remainder of this thesis.

1.1 Circuit Design Methodology

As contemporary system-on-chip (SoC) designers push for ever more functionality and performance from their designs, they are finding that additional design complexity and bleeding-edge process nodes are leading to an increase in post-silicon activity in the design-phase (verification, testing, and diagnosis) with most of it occurring around AMS components and their integration. Designs are undergoing increasing numbers of silicon re-spins due to problems originating in both design and manufacturing errors. Troubleshooting between respins yields varying degrees of information to be fed-back to designers and process engineers [1, 5].

In current practice, AMS testing and debugging techniques tend to be ad-hoc and so result in long lead-times, high human effort, and low tool-reuse. Additionally, successful industry practices are heavily guarded intellectual property, and are not routinely disclosed within

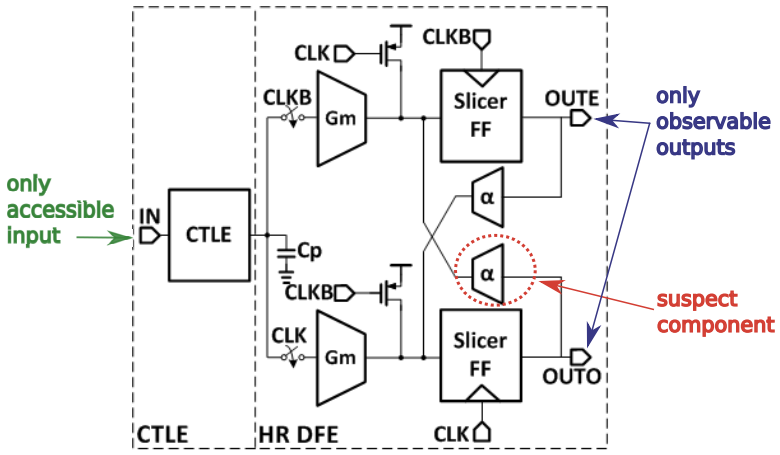

Figure 1.1: A Modern Fault Localization Problem visualized in a high-speed I/O Discrete-time Linear Equalizer [6]

the community.

Traditional test stimulus design is predicated upon reliable knowledge of both the design and manufacture of the DUT as well as presumption of realistic fault models and/or bug manifestations. In cases where the design is thoroughly known and accurate models of all behaviors exist, hierarchical simulation models can be used to probabilistically target failure modes (or process variation or bugs) and generate tests which might best probe those vulnerabilities and carry information to an observable output.

At present, there exist varieties of contexts in which either the abstraction model, composite model, fabricated DUT, or some combination cannot be relied upon to be accurate or dynamically consistent representations of the same thing. Disagreement between DUT and model or high- and low-level models can arise as a result of process variation, design tool limitations, or model shortcomings.

Further complicating matters, most modern test paradigms rely on a means of test pattern generation which itself requires a trustworthy model. In situations of post-silicon validation, the device's simulation model cannot be relied upon as a vehicle for test design because physical realization of the device may introduce behaviors which the model does not include (e.g. ground bounce, negative-bias temperature instability, positive-bias

temperature instability, radiant leakage, cross-module coupling, power supply limitations, etc.).

Additionally, in large AMS designs, observation and control of internal signals and states is rarely complete; this is the case in silicon and similar conditions can arise in simulation due to practical constraints on data collection. Even the most novel design-for-test (DFT) techniques cannot provide access to all internal nodes of an AMS chip (Figure 1.1). The inability to directly stimulate and observe embedded module inputs and outputs is a tremendous impediment to detecting undesired system behavior and identifying the origin of design or manufacture errors.

What's required is a comprehensive framework for establishing and quantifying trust at low levels of the model hierarchy which can be measured and propagated across abstraction layers and through composite simulations.

1.2 Basic Definitions

First, let me define my use of the word "system." In the context of this thesis, we use the term "system" to refer various analog and mixed-signal (AMS) electrical circuits. Formally, all of these circuits are surjective mathematical functions. That is to say, every unique input (within some domain) will map to an output, and two different inputs may map to the same output.

Next, let us draw a distinction between the terms "stimulus" and "input" in the formal context. The term "stimulus" refers to a single or a set of vectors, each of which will be applied to a system input over the course of some time. In common parlance, a "stimulus" is simply a waveform. An "input" in the formal context refers to a piece of information in the fundamental unit of uniqueness in the input space. For example, if our system is "static" or "memoryless," any scalar value applied at the input port of the system could potentially result in a unique output. Therefore any scalar value becomes "an input" to that system. On the other hand, if our system has 1 unit of memory (that is to say it retains knowledge

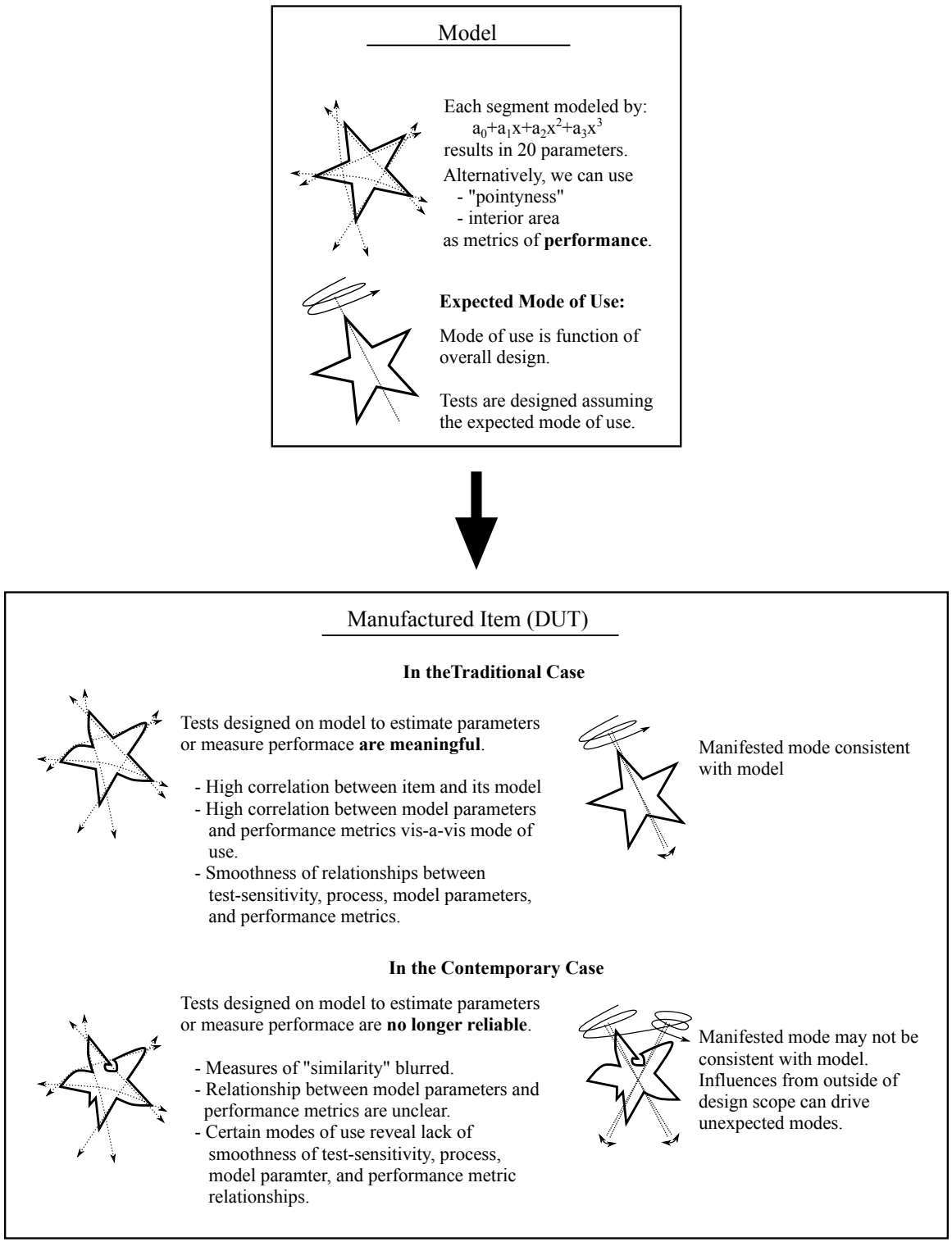

Figure 1.2: The Contemporary Post-Silicon Validation Problem As Analogy

of the previous input value), "an input" would refer to both the present and previous scalar values presented at the system's input port (a vector of length 2). And so, a "stimulus" is actually the composition of many inputs to the system arranged in a particular fashion in time.

In this work, we conduct experiments by applying a stimulus to a system under test. Over the duration of the experiment, we record the output(s) of the system. A sequence of observations made during a single experiment is referred to as the "response" of the system. In the same way that a stimulus is a composition of inputs, a response is a composition of outputs. And so we have sequences of inputs composed into a stimulus which are applied to a system which, by some mathematical function, produces a response which is recorded as a sequence of outputs. Together, we can compose our stimuli and our responses and refer to the collection as "experimental data."

1.3 The Relative Merits of Stimuli

It is the job of a stimulus to reveal information about the system, a "good" stimulus revealing a large amount of information. What we see when we look at a system's response is only a transformed version of the same information present in the stimulus. And so, a high-complexity stimulus doesn't necessarily hold any advantage over a simple one.

Let us look at a simple example. Figure 1.3 depicts three different stimuli within the range (-1,1) which are applied to a (memoryless) function. Comparing the stimuli (in the first row), we see that the sinusoid is highly structured, but lacks complexity. In contrast, both the Gaussian and uniform white noise lack structure but are rich in complexity. In the second row, we see the system's response to each stimulus; we note that the range of data expressed in all cases in comparable. In the third row, we compare the shapes of the data distributions in the stimuli and in the responses. One measure of the system is the degree to which information is transformed by passing through [7]. In the third row, we can see the underlying function emerge in a plot of output vs. input. Shaded vertical bars correspond to

increased observational density in the input space, and shaded horizontal bars correspond to increased observational density in the output space.

If we were to attempt to model the function, which would be our preferred method by which to sample the function? The sinusoidal input results in a large number of samples coming from the extremes of the function's range and very few samples from its interior. The normally distributed noise, however, results in heavy sampling from the portion of the function's range near the origin, where it is mostly linear. Being approximately linear, there is little transformation of information in this region, explaining why the distribution of output data so closely matches that of input data. The uniformly distributed noise, generates samples coming chiefly from the extremes, though it's not as imbalanced as either the sinusoid or the Gaussian noise.

In this case, since we've discovered that most of the function's curvature lies in those regions most heavily sampled by the uniform white noise, we would prefer it. To use the Gaussian noise to glean the same amount of information revealed by the uniform white noise, one would have to conduct several experiments. And so, the Gaussian white noise stimulus lacks economy in this case. It should again be noted that we can only draw conclusions about efficiency after establishing a basic understanding of the location of features in the function space.

Let us now look at a more complex example. Figure 1.4 summarizes experimentation on a system which has memory. In the first row of plots we see our familiar input stimuli decomposed into sets of two-dimensional input data. We note that the sinusoidal stimulus only covers a thin slice of the input space parameterized as $X \coloneqq \{x(t), ; x(t-\tau)\}$. In the second row of plots, we see the responses of the system to the different stimuli. Here we note the diversity in response amplitudes. For a better understanding of why these responses look so different from those in the memoryless case, we turn to frequency domain representations of the data. In the third row of Figure 1.4 , we can compare the distributions of energy across frequency for all stimuli and their corresponding responses. One

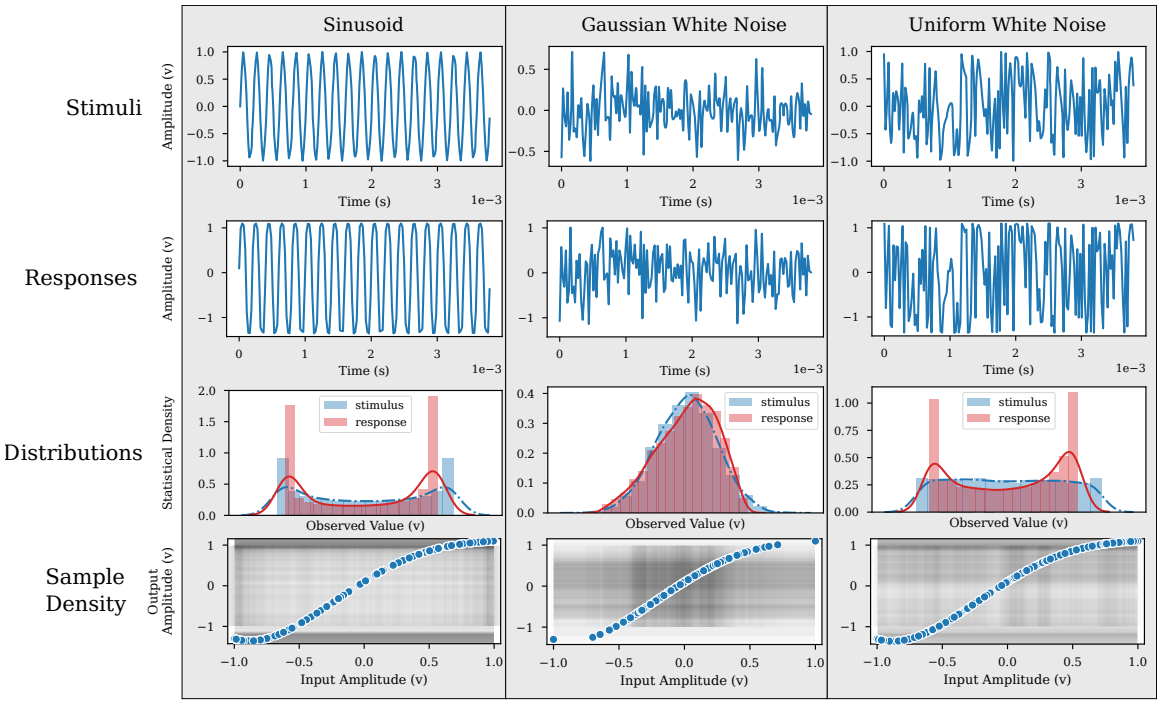

Figure 1.3: Comparative Study of Stimuli for a Memoryless System: Observations of informational content revealed by three stimuli, a sinusoid, Gaussian white noise, and uniform white noise.

should note that in a memory-having system modeled under an assumption of memorylessness, the output does not appear to be a function of the input (strictly speaking). One also observes that the different input distributions result in different sample densities in the input space and output space. We see the system nonlinearity manifested in the presence of additional high-frequency energy relatively speaking.

The fourth, fifth, and sixth rows all present the results of solving a regularized least-squares (RLS) problem with our data under varying assumptions of input dimension. In the fourth row, we model the system as memoryless, and we note that no experimental data is sufficient to train a successful model. In the fifth row, we model the system as having memory = 1, and we find that the RLS model trained on sinusoidal data can provide reasonable prediction accuracy, but the data resulting from Gaussian and uniform noise stimuli is insufficient. It is not until we reach a model complexity with a memory of 64 data points, in the sixth row, that the data from the Gaussian and uniform stimuli can be put

to use.

Hopefully these examples have brought to light some of the dimensions of the stimulus design space and some of the fundamental trade-offs. The principal considerations are:

1. The time duration of the stimulus: though constant in our examples, a longer test can potentially reveal more information than a shorter one.

2. The complexity of the stimulus: This correlates with the difficulty of interpreting results.

3. Stimulus coverage of input space: if we measure the domain of our test in the input space, we must consider how our stimulus covers that space.

4. Sample distribution in output space: test economy depends on how many repeated measurements are made.

1.4 Fundamentals of Testing

For as long as there has been experimentation, there has been testing. In a piece of work published in 1873, engineers measure telegraph circuit performance by counting the number of Morse coded characters successfully received [8]. An early paper on transistors lays out a detailed processes for obtaining "load lines" of semiconductors [9]. Early papers on mechanical control systems lay out procedures for measuring and comparing performance using step function inputs [10]. Testing has evolved implicitly alongside other major trajectories in the field.

Formal approaches to "testing" in its own right began to emerge in the late 1950s along with the introduction of electrical computing machinery [11]. From these origins, an entire field has emerged and continues to flourish, focused upon ensuring the functionality of logical computing machinery. Notable contributions include Roth and Bouricius' DALG

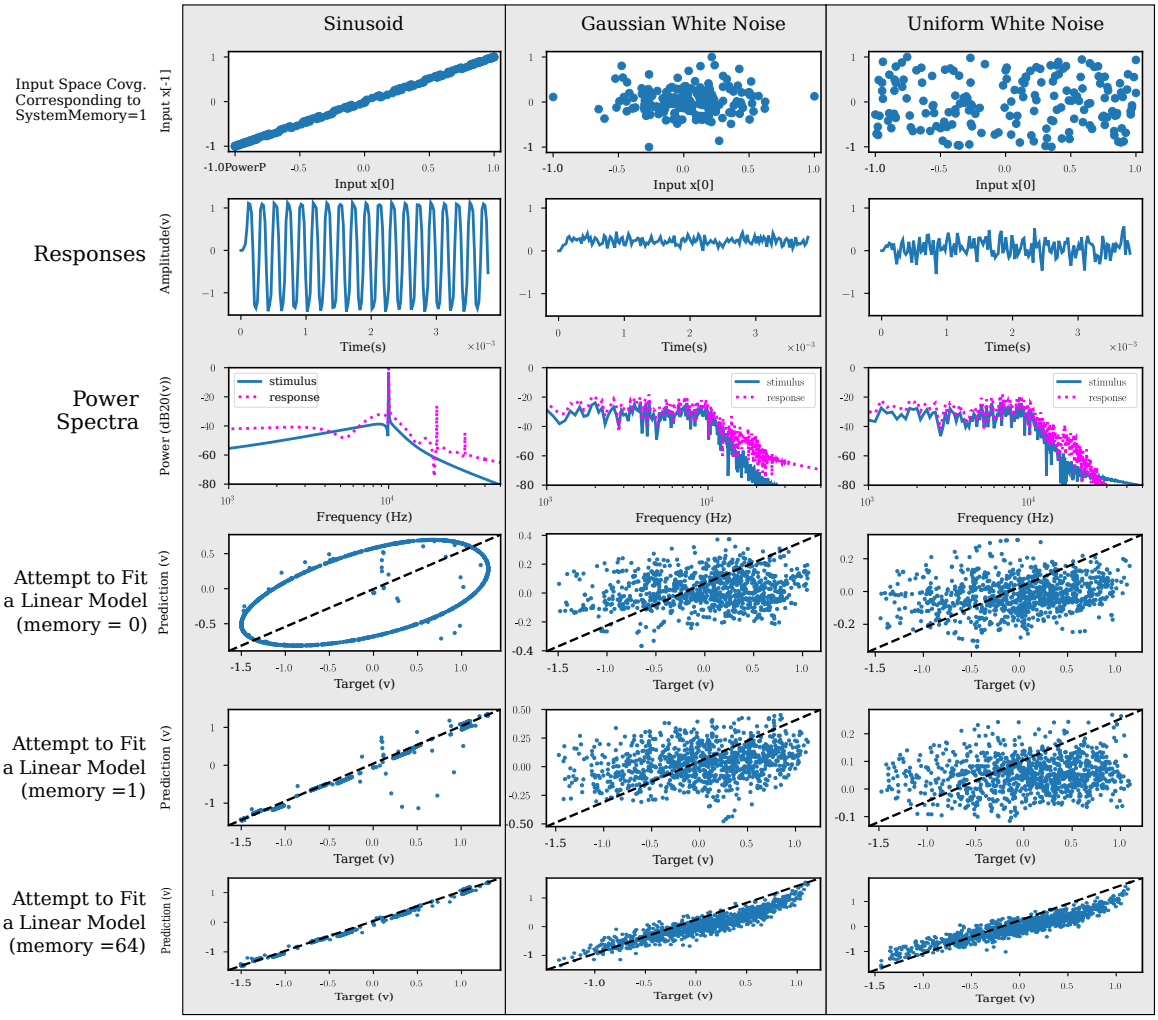

Figure 1.4: Comparative Study of a System with Memory: Observations of informational content in experimental data using sinusoidal, Gaussian white noise, and uniform white noise.

[12, 13], Chappell's LAMP [14], and Goel's PODEM algorithms [15]. From these pieces of work came formal notions of fault models, test coverage, test escape, and yield loss.

Unfortunately, the rapidly maturing study of testing bore very little fruit for AMS system designers and manufacturers. Approaches to testing analog and radio-frequency (RF) systems have not evolved much from the ad hoc techniques seen in Sauty's telegraphy paper [8]. For example, the "two tone test" described in a 1930 paper discussing radio receiver testing is still the de facto test for measuring RF system linearity and is used in work as recent as 2017 [16, 17]. A panel at the 2015 European Test Symposium asks "why [is analog test] still '*a la mode*'after more than 25 years of research?" [18]

Presently, AMS SoCs require the integration of digital, analog, and RF components onto a single die, and so novel testing techniques must be developed to meet the expectations of both digital and analog engineers in testing for the presence of physical defects as well as conceptual, design defects. Even if one believes the existing AMS testing paradigm is mature and complete, extensive work must be done even if only to extend that paradigm to AMS components buried deep within a System-on-chip (SoC).

1.4.1 Defect-based Testing

Defect-based testing holds the allure of quick, directed tests that target commonly observed defect mechanisms, forsaking rigor for speed. It requires an enumerated dictionary of failure modes at the beginning; then it creates a minimal test sensitive to a maximal portion of the dictionary [19].

In [20], the authors use an impulsive stimulus to excite various faulty filters. After training, an artificial neural network (ANN) classifies systems as "pass" or "fail." Methods were introduced by [21] wherein the authors explore circuit nodes most sensitive to a fault set and find a multi-tone stimulus that can be used to increase net detectability of the fault set.

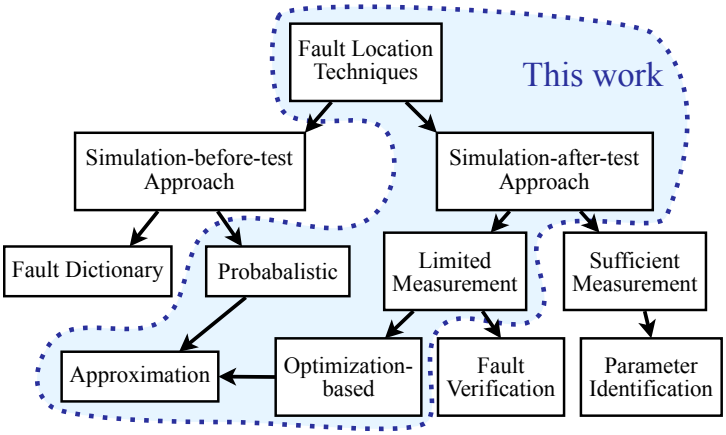

Figure 1.5: The Fault Location Technique Universe [22]

1.4.2 Diagnosis

Diagnosis, like defect-based testing, is concerned with local phenomena on the die. Though defect-based testing merely detects the presence of any single fault, diagnosis attempts to infer *which* fault is present. Figure 1.5 shows the general classes of methodologies as presented by [22].

The bulk of work in the field of failure diagnosis can be found in either of two categories of literature: those that frame fault localization as a "gray-box" system identification problem, and those which frame it as a classification problem.

1.4.3 Analytical methods

Analytical methods attempt to diagnose the location of faults through formal mathematical approaches, typically by solving optimization problems to calculate parameter values. They require "well behaved" and accurate component models.

In [23], the authors detect and diagnose relatively small deviations in component values of linearized circuits using linear programming methods to evaluate the feasibility of single-parameter deviations leading to observed behavior. In [24], the authors present a methodology for diagnosing specification violations by through alternate test inferences of parameter values in analog/mixed-signal circuits. Similarly, techniques such as those pre-

sented in [23] and [25–29] leverage information from observed outputs to back-calculate model parameter values when the model is known to be accurate.

1.4.4 Classification Methods

Most modern contributions employ machine-learning algorithms to classify failed systems based on the similarity of observed behaviors to a set of prototypical faulty behaviors. The inherent shortcoming of these approaches is the necessity of a training set of failed systems for the classifier to use. Arrival at a training set for a large contemporary design, however, is no easy task. One must enumerate a set of "feasible" faults which, for systems of even modest size, will result in intractable numbers. And so commonly, one must then intelligently sample the relevant design and fault space to reduce the number of simulations required to a reasonable size [19, 30, 31].

[20] uses a neural net classifier to diagnose faults based on systems' impulse response rather than an optimized stimulus. Similarly, fuzzy classifiers [32], white noise stimulus [33], and multi-tone stimulus [34] can be used to perform fault diagnosis provided the fault in question is represented in the training set of faulty devices, something very difficult to guarantee. In [35], a methodology for sampling from a fault universe (parametric faults) is presented which leverages the fact that many faults have similar "syndromes" which can be clustered. [24] and [36] use a genetic algorithm to optimize a piecewise linear test stimulus before using regression modeling to relate the device parameters to the measurements made on the DUT. It was shown that from this relationship, a cause-and-effect analysis could be used to determine the parameter whose variation led to the system specification violation. In lieu of an analytical identification of the faulty system, [37] presents a heuristic best-guess approximation of the buggy subsystem based on feasibility estimation.

1.5 Alternate Testing

The one major advancement to emerge from the formal study of testing which *does* apply to AMS systems is the Alternate Test methodology. Introduced with the concept of analog signatures by Nagi et al. in [38], alternate testing is the practice of drawing conclusions about high-level system performance based on statistical inferences made from a minimal set of measurements. It is an *alternative* to performing a battery of tests, each of which measures a single performance specification. As an example, alternate testing can enable the classification of parts into "pass" and "fail" categories through analysis of a single analog response waveform or DC value [39].

The concepts of alternate testing really took off when coupled with advanced computational tools like Multivariate Adaptive Regression Splines (MARS) [36, 40], support vector machine (SVM) classifiers [41], wavelet-packet decompositions [42], and ANNs [33]. Primarily useful in decreasing cost, complexity, and duration of production test [43, 44], the concepts also proved useful for post-manufacture tuning (e.g. laser-trimming circuit elements or blowing fuses)[45, 46], and compact built-in self-test (BIST) schemes [47].

Finally, alternate test has enabled the automated generation of analog test stimuli which can be optimized for inference of particular high-level characteristics. The authors in [48] have performed genetic algorithm based stimulus generation where each stimulus's fitness is measured by the ability of a non-linear solver to correctly back-calculate model parameters based on the DUT's response. In [49], the authors use a genetic algorithm to find a globally optimal transient stimulus for both fault detection (pass/fail) and fault diagnosis (which parameter has varied).

All of these methods are highly model dependent, i.e. all test strategies are generated *a priori* on a presumptive model of the system. For instance, the stimuli generated in [50] will provide back-calculability of model parameters only if the model used in stimulus generation is accurate and complete. Neither model accuracy nor completeness can be

assumed in the most contemporary test applications.

1.6 The Validation Problem Formalized

To provide an entry point to validation, we turn to [4]. Here, the authors provide an analytical definition of model validation:

$$\text{argmax}_\theta \, \mathcal{L}(\theta|x) < k \implies \text{invalid model} \tag{1.1}$$

where the maximum likelihood estimate maximum likelihood estimate (MLE) parameterization is compared to a chosen threshold value, k. Equation 1.1 suggests that if there is no parameterization of our model which sufficiently explains the data we've been given, then we reject validity of our model. The authors of [4] go to great lengths to underscore the idea that we **cannot prove validity**, an idea I took to heart.

To point out some of the complications in adopting this formalization as our premise, I make the following observations:

1. The formalization does not address how one should obtain the data, x.
2. There is no explanation as to choice of threshold, k.
3. The formalization assumes that the model parameter space is tractable.
4. The formalization assumes that a likelihood function can either be analytically or computationally evaluated.

Given a minimal set of assumptions and some data, one can reduce the solution of the problem to a nonconvex optimization problem: find the maximum likelihood parameterization of your model, and from there determine whether it meets a chosen threshold for validity. If one finds that the criterion is not satisfied and rejects validity, the work is done; If validity cannot be rejected, what then? Contemporary system designers routinely handle components with hundreds and thousands or parameters, whose nonlinearities require

evaluation through simulation rather than analytical reduction. As such, solving even the MLE problem becomes a costly proposition in terms of computation, and additionally, the component designer must himself generate the data with which to validate his behavioral model.

1.7 Post-Silicon Validation

Post-silicon validation of mixed-signal/RF circuits centers around the fact that neither model accuracy nor completeness can be assumed in the most contemporary designs; neither can bug or fault models be assumed [1, 51, 52].

Post-silicon validation presents the key challenges:

1. What test stimuli should be applied to the device-under-validation (DUV) in order to expose any behavioral differences between the silicon DUV and its model which may exist?

2. Is a discrepancy attributable to incompleteness of design specification, design error, fabrication error, or inaccuracy of model?

3. How can the behavioral model of the DUV be modified to best capture any behavioral discrepancies between the DUV and its model to enable design rework?

An ideal validation test would stress the DUV over its entire input space and vary its operating environment (supply levels, temperature, output loading, etc.) over *its* entire space while comparing the behavior of the silicon to a perfect representation of the designers' intent. Such an approach is not feasible. Although the set of possible inputs to an entire digital system (e.g. a personal computer) is also impractically large, tests of hierarchical subsystems can be bounded, memoryless combinational blocks can be tested in isolation, and novel DFT techniques can be employed [53]. Though much work has been done toward formal validation of large VLSI designs [54–57], AMS validation is an infant field.

Validation of AMS systems is difficult primarily due to vastly increased design sensitivities and increased dynamic ranges and is compounded by difficulties in stimulating and probing high-frequency/high-impedance internal nodes. Only recently, has there been progress in post-silicon validation and debug of mixed-signal/RF circuits and systems. In [58], a test generation approach is proposed that directly compares the observed DUV response to its model's response and stochastically optimizes a stimulus to expose differences between the two. In [59], an optimized test waveform is applied to the DUV and its model, and it is seen that in the presence of unexpected DUV behaviors, no degree of parametric manipulation of the model can reduce residual error below a lower limit. This work makes no assumption about failure mechanisms, but assumes that input vs. output behavior is invertible. When multiple inputs as well as manufacturing process variation need to be accounted for, such inversion cannot be assumed. Building on this work, a validation-failure diagnosis approach is proposed in [37], and a model adaptation technique is proposed in [60].

1.8 Fault Modeling

One element of the formal study of testing which applies to modern testing problems is the fault model. A classical example from digital testing is the "stuck-at" fault: a fault model proposed by Armstrong which generalizes all manner of physical defect within a gate by fixing a faulty gate model's output at one or the other logical value [61]. The stuck-at model enabled a volume of work; some which revealed that it's possible for a fault to exist yet have it's presence completely masked [15]. This begs the question of what is means for a *system* to be faulty because though it may contain a fault, a system's behavior may be indistinguishable from a fault-free system. Sunter addresses this in [19] for AMS systems by defining a faulty system as one whose performance violates at least one system-level specification.

Commonly assumed transistor fault models include gate-to-drain, gate-to-source, and

drain-to-source short-circuit defects as well as gate, drain, and source open-circuit defects. Fabricated analog circuits pose a more nuanced problem: everything exists on a continuum, including opens and shorts. A traditional distinction drawn in papers on manufacturing test is that between "hard" and "soft" faults. Hard faults are those that model catastrophic device failures while soft faults are those arising due to variation of at least one parameter to sufficient degree [22, 62]. A tremendous amount of work has been done which addresses these two categories independently. For example, Milor and Zhang go after catastrophic failures using alternate tests and statistical techniques [39, 63] while others have focused on the detection of parametric faults [30, 62, 64–68]. Sunter and Stratigopoulos have proposed techniques for deriving coverage metrics for tests addressing faults of both kind in [69] and [30]. When dealing with nonlinear circuits, it's not clear that good coverage of "hard" and "soft" faults translates to coverage of "medium" faults.

To distinguish between hard- and soft-faults belies an underlying commonality: a hard fault is but an extreme degree of a soft fault. Kundert, Mitra and others acknowledge that there exists a larger category beyond "hard" and "soft" in which one would find systems which fail due to crosstalk between components, power supply bounce, clock feed-through, oscillator lock-in and pull, or radiation [1, 51, 55, 56, 70], thus reformulating the classification of failures into two categories: "parametric faults" and "everything else." In contemporary validation, detection, and diagnosis testing, the "everything else" category requires equal attention, though the current body of knowledge doesn't yet explain how to provide it.

1.9 Dynamics Modeling

In this work, we are primarily dealing with problems existing between a detailed model and an incorrectly specified reduced-order abstraction model. A great deal of focus is given to the analysis and decompilation of system dynamics in effort to build and/or refine hypothetical reduced order models from observed dynamics – given the defficient abstraction

model.

There has been a great deal of work in the domains of model-order reduction (MOR) and automated model extraction. For example, work in [71–75] all propose tools for extracting from an explicit set of differential equations a reduced-order approximate representation of the same system in a lower-dimensional phase-space. Generally, Krylov subspace methods are employed and the techniques are applied to linear or *weakly nonlinear* systems through various piecewise assemblages of linear models.

In [76], the authors present an extension of the projection of the ODE system by performing non-linear projections onto manifolds in high-dimensions as a way to further reduce redundancy observed in linear projections of non-linear systems.

In [77], the authors present a more heuristic, circuit-designers approach for accomplishing the same goal. In that work, the authors were able to achieve a reduction in the number of model variables on the order of 50 percent, and a reduction in the number of model parameters on the order of 1000 percent, leading to simulation-time speedup of on the order of 10x.

Similarly, the authors of [78] systematically null equation elements contributing to each row of the system's ODE, thereby removing low-consequence dynamical factors.

Other researchers have proposed abstraction models which step outside of the numerical ODE/PDE simulation paradigm. For example, in [79], the authors represent trajectories through a system's state-space in the form of entries in lookup tables and re-structure analog dynamics as a series of sequention table look-ups, thus enabling digital verification techniques to be applied to the booleanized circuit. Additionally, the authors of [80], in order to preserve the stability of the system, build-up from scratch a new polynomial-basis ODE representation of the system, solving for parameters which meet accuracy criterion as they go.

Also of note: a patent was issued in 2002 in which the authors present a modeling technique which resides entirely in the context of an FPGA wherein individual executables

are (optinally dynamically compiled) and dynamically linked by a solver depending on the state of the simulation. In this way it functions as a dynamical piecewise (non-) linear simulation code execution environment.

After synthesizing a new reduced-order model, they must be checked. Nearly all existing literature or MOR uses a fixed set of a priori identified randomly chosen stimuli, exhaustive enumeration of discretized stimuli, or fixed sets of *likely* stimuli to evaluate the performance of the model. The measure of perforance is almost always mean square error (MSE). The sortcomings of MSE as a performance metric in this context are addressed in Chapter 4 . Some literature, such as [81], address model verification in a more dynamic way.

In [81], authors use a meta-language known as "Bounded Linear Temporal Logic" which operates on extracted properties of time-domain observations to project them into a boolean space. This logic provides a way to formalize performance properties like "gain" over arbitrary trajectories in a system's state-space and to bound acceptible ranges to be considered "valid." The authors then employ a Baysian sampling scheme to run simulations and collect evidence on whether properties are consistent between models.

CHAPTER 2

OPTIMIZATION-BASED APPROACHES

2.1 Hardware-based Guided Stochastic Test Stimulus Generation

In this section, I introduce the RAVAGE algorithm (from "random," "validation," and "generation"), which uses a fabricated AMS hardware system to generate its own test stimuli to be used for post-silicon validation of mixed-signal systems [58]. The approach of RAVAGE is new in that no assumptions are made beforehand about the nature of the anomalies which the test seeks out within the DUT; but rather, the stimulus is generated using the DUT itself. The objective of RAVAGE is to maximize the magnitude of any behavioral differences between the DUT (hardware) and its behavioral model (software) observable in their responses to the same stimulus. Stochastic search methods are used since the exact nature of any behavioral anomaly in the DUT cannot be known a priori. Once a difference is observed, model parameters are tuned using nonlinear optimization algorithms in attempt to resovle the difference in responses and the process (test generation $\rightarrow$ tuning) is repeated. If a residual error remains at the end of this process that is larger than a predetermined threshold, then it is concluded that the DUT contains unknown and possibly malicious behaviors that need further investigation. Experimental results on an RF system (hardware) are presented to prove feasibility of the proposed technique.

2.1.1 Algorithm

The RAVAGE stimulus generation engine provides a means of performing *on-the-fly* test stimulus generation which is a function of not only the model but also the DUT itself. RAVAGE stochastically searches the operating space of a single DUT with the goal of stressing it to the point that equivalence to its model can either be rejected, or be asserted

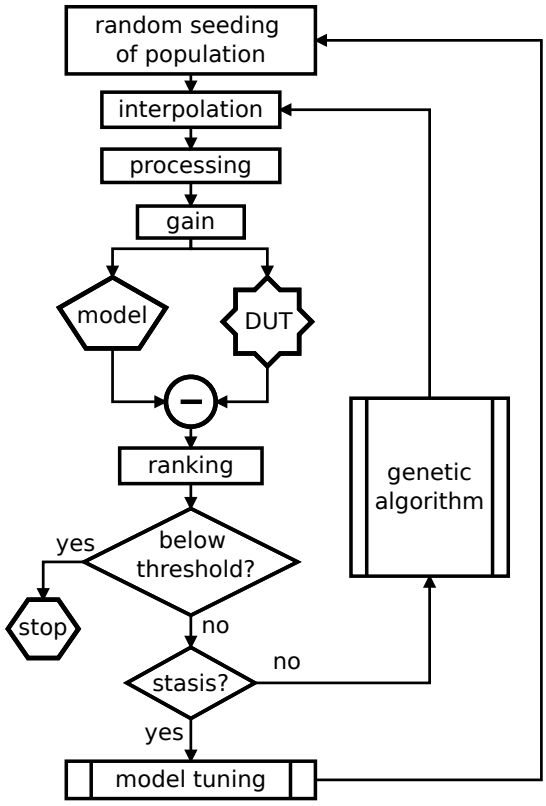

Figure 2.1: RAVAGE System Overview

with some certainty. Whether or not the process corner of the DUT is known, whether or not the DUT is suspected of errant behaviors, RAVAGE may be used to validate (or invalidate) a DUT against its model.

The RAVAGE algorithm begins with a population of arbitrary band-limited stimuli and employs a genetic algorithm to evolve the population (within amplitude and frequency constraints) in such a way that it best excites behaviors in the DUT that the DUT's model does not exhibit. Whether due to structural or topological inadequacy of the model (equivalent to unexpected circuitry in the DUT), lack of complexity in the model, or process variation in the DUT 0(equivalent to incorrect model parameters), RAVAGE will favor those stimuli which best expose differences, regardless of their origin. If stimulus performance improvement becomes static, RAVAGE attempts to tune the model to capture the observed discrepancy. The stimuli used for tuning are then put aside, the entire population is reinitialized with random seeds, and the process repeats. If RAVAGE can tune the model sufficiently,

such that error power never exceeds a threshold, then the model can be treated as complete, both behaviorally and parametrically. RAVAGE can accept a model of any complexity simulated in Matlab or Spectre and can interface with Spectre to tune spice models.

Stimulus Implementation

A population, G, of i individual stimuli is created. Each individual stimulus, s_i, is initialized to randomly generated white Gaussian noise. The length of s_i is determined by the desired bandwidth of excitation, and they are considered to be sampled at the Nyquist rate. These stimuli will be undergoing mutation in the frequency domain, and the applied stimuli, S_i are derived from this reference population before each use. To arrive at S_i, each s_i undergoes low-pass interpolation, additional low-pass filtering, application of an envelope, DC removal, and normalization to the desired amplitude. S_i and s_i are kept at differing sample rates so that stimuli can be applied using high sampling rate arbitrary waveform generators (AWGs) while allowing the genetic algorithm to operate only over the bandwidth of interest. The interpolation ratio is decided depending on the maximum desired frequency resolution of the stimulus and the sampling rate of the AWG. Interpolation is performed by injecting L zero-samples between each original sample and subsequently performing low-pass FIR filtering. An amplitude mask is then applied to the stimulus, to diminish out-of-band high frequencies resulting from instantaneous envelope transitions. A hyperbolic tangent sigmoid vector is multiplied with the first and last w samples:

$$s'(t) = \begin{cases} \dfrac{1}{1+e^{-k\cdot s(t)+5}}, & \text{for } 1 \le t \le w \quad (2.1a) \\ \dfrac{1}{1+e^{k\cdot s(t)-5}}, & \text{for } T-w \le t \le T \quad (2.1b) \\ s(t), & \text{otherwise} \quad (2.1c) \end{cases}$$

where k controls the rise-time of the envelope.

Additional low-pass filtering is performed using a Chebyshev(I) low-pass IIR filter to remove any aliasing remnants from the interpolation operation and envelope application. Finally, DC is removed, and the signal is normalized, guaranteeing to reach but not exceed a user-specified amplitude. Both the low-pass filtering and gain of the stimuli are designed based on the expected system specifications. Each member of the population S_i is then zero-padded and concatenated for application to the DUT. The concatenated stimuli are sent through the AWG and the DUT, and the response is captured by a high-speed digitizer. At the same time, S is used in transient simulation on the model of the DUT.

Fitness Function, Biasing, and Error Metrics

The bias provided by an effective fitness function is critical to the the evolution of the stimuli. RAVAGE's fitness function for evolution is as follows: The DFT of the DUT's response, the model's response, and the applied stimulus, S_i, is calculated. The power-specral density (PSD) is then computed for each, and the model's PSD, P_m is subtracted from the DUT's, P_d. This error power vector, P_e represents output power differences between the DUT and its model. For the remainder of fitness and error metric calculation, only a predetermined bandwidth of interest (BOI) is considered. The L_2 norm of this vector over the BOI is kept as a measure of each stimulus' effectiveness. The stimuli that yield the largest error are preserved from generation to generation. This error metric, however, does not function as the fitness function for genetic evolution. Instead, the population is groomed not only to maximize error power, but is penalized for yielding a spectrally sparse error power and is rewarded as stimulus power, P_s decreases. So, in addition to the scalar error power metric, a fitness value, $F(S_i)$, is calculated for each stimulus:

$$F(S_i) = \frac{\|P_{e_i}\|_2 D(P_{e_i})}{P_{e_i}} \tag{2.2}$$

Where $D(P)$ gives a scalar representation of the spectral "richness" of a PSD vector on the

range $[0, 1]$ (N here is the number of frequency buckets over the bandwidth of interest):

$$D(P_{e_i}) = N \left\| \frac{P_{e_i}}{max(P_{e_i})} \right\|_2 \tag{2.3}$$

The $\|P_{e_i}\|_2$ of each stimulus is termed *total error power*. The reference stimuli are then ranked. The top η performers in scalar error power are ranked first, followed by the rest in order of descending fitness. Additionally, the greatest total error power of each generation is monitored, and if it is seen to stagnate for a period, RAVAGE is determined to have reached *stasis* and will either begin model tuning or terminate. Otherwise, genetic mutation is performed over the population, and the process repeats.

Population Genetics

The absolute performance of a stimulus is not measured by the fitness function that guides its evolution, but rather it's measured by the total error power the stimulus yields. The best η stimuli (according to the previously explained rankings) are preserved and are subsequently known as *elite* stimuli. The elite, along with the next ϵ are then considered *survivors*. The remainder are destroyed to make room for new stimuli (total population size i is constant). New stimuli are created in three ways (Fig.2.2):

1. All of the survivors mutate, including the elite (though the elite will propagate unmutated as well), to yield *mutants*.

2. Pairs of survivors are arbitrarily selected to procreate. Their DFTs are split at a random frequency and are spliced together to create the *children*'s DFT which subsequently undergoes an inverse DFT.

3. *Alien offspring* are introduced which are the product of procreation of one survivor with an *alien* stimulus (a new white gausian stimulus) in a similar manner to 2).

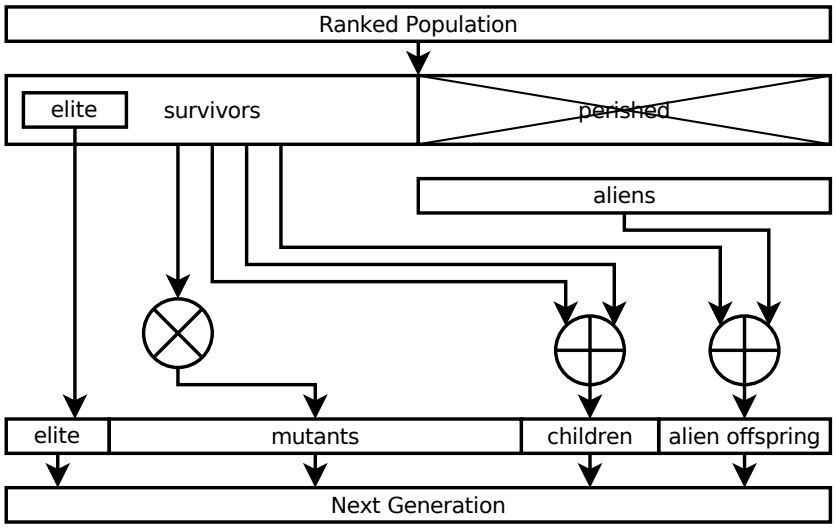

Figure 2.2: Population Heredity

Mutation is performed over the survivors by separately and arbitrarily disturbing the magnitude and phase of each stimulus' DFT before taking the inverse-DFT. Each amplitude is scaled randomly based upon a Gaussian distribution centered around unity and each phase is mutated by adding random phase based upon a Gaussian distribution centered at zero. In each case, the distributions' σ is globally controlled and termed *mutation factor*. This new generation of reference stimuli are processed, as previously described, before application to the DUT and model.

Model Tuning

After some number of generations, successive improvement in total error power between generations will reduce and approach zero. The genetic algorithm has reached a local maximum, and will likely not begin to improve again. This is a good opportunity to take advantage of the current population of stimuli to tune the model parameters to reduce the total error power. The model (which exposes some number of "knobs" to the solver, thereby enabling use of models of any detail), a subset of high performing stimuli, and their corresponding DUT responses are passed to a non-linear least squares solver. The model parameters which the solver finds to yield minimal total error power are selected as can-

didate parameters. The candidate parameters' performance is compared to performance of the previous parameters using the remainder of the current population, and if improved, the new parameters are kept.

2.1.2 Completed Experiments

Test Equipment

RAVAGE was implemented in Matlab 2011b on a conventional Dell quad-core Xenon workstation. National Instruments LabView is run concurrently with Matlab and is responsible for communication with the test instruments. Matlab scripts control states of variables within a Matlab-VI in LabView, thereby controlling the test instrumentation. A National Instruments PXI-1073e chassis houses the AWG and digitizers, respectively a PXI-5412 and PXI-5105. National Instruments the PXI-5412 AWGs' square-wave output can be trusted to 5 MHz, and the PXI-5105 is operated with its 24 MHz anti-aliasing filer in place. The sample clock PLLs in the two cards are both referenced to the 10 MHz backplane synchronization signal, and a sample time accuracy of much less than one sample is achieved at 60 MHz sampling rates. More important than triggering delays, however, is the relative agreement of sample clocks. Because processing is occurring entirely in the frequency domain, any frequency translation resulting from disagreeing sample clocks is hazardous.

DUT=wire , MODEL=wire : In this experiment, cables directly connect the output of an arbitrary-waveform generator (AWG) to the input of a digitizer and the system is modeled by an ideal wire. A Photo of *Experiment 1*'s hardware setup is provided in Fig.2.3. The system is modeled by an ideal wire. Using a maximum stimulus bandwidth of 5 MHz, and an error BOI of 18 MHz, RAVAGE was allowed to run for 853 generations. This experiment was run primarily to determine the effective baseline total error power. Fig.2.4 shows the fitness and total error power over 400 generations. The stimulus that achieved

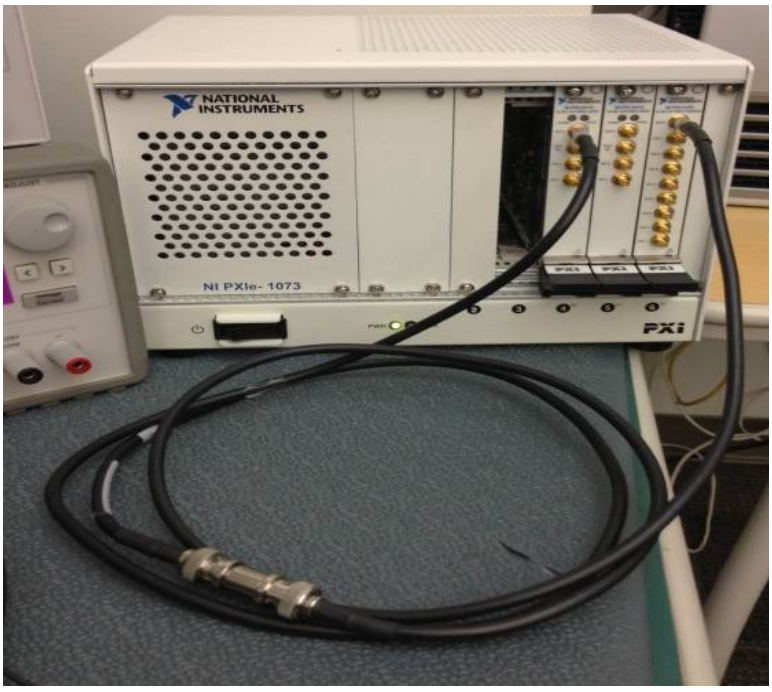

Figure 2.3: Experiment 1 DUT

the maximum error is shown in Fig.2.5.

DUT=up/down mixers , MODEL=wire : In this experiment, a pair of Maxim MAX2039 high-linearity up- and down-conversion mixers are inserted between the AWG and digitizer but the model remains an ideal wire. Care was taken to ensure phase coherence at each mixer's LO input. Fig.2.6 provides a photograph of the MAX2039s while under test. The model in this experiment was left undisturbed from Experiment *exp:ravage1*. Using a maximum stimulus bandwidth of 5 MHz, and an error BOI of 18 MHz, RAVAGE ran for 623 generations. Fig.2.7 shows the fitness and total error power of the population through 60 generations of *Experiment 2.1.2*, and Fig.2.9 shows a time-domain plot of the most effective stimulus. *Experiments 2.1.2 and 2.1.2* collectively give RAVAGE an effective pass/fail discrimination SNR of 32.5 dB based only on the nonidealities exhibited by Maxim's high linearity converters.

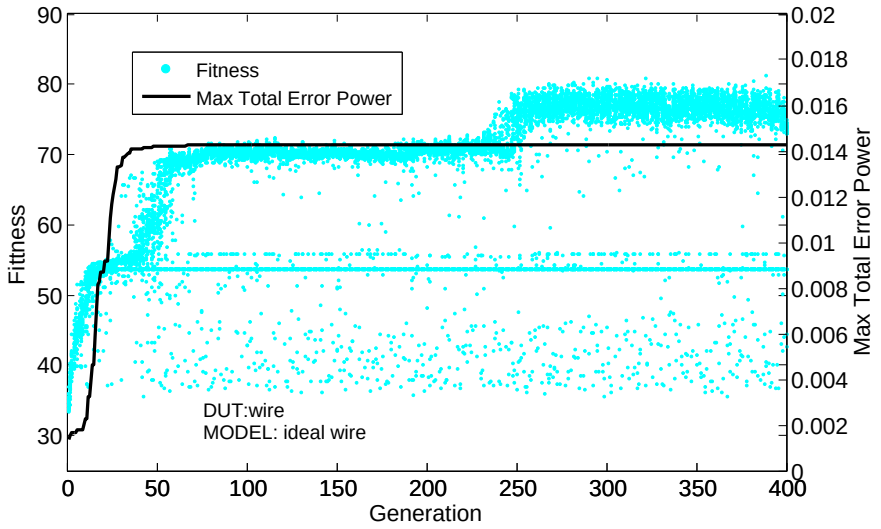

Figure 2.4: Experiment 2.1.2(a) Results

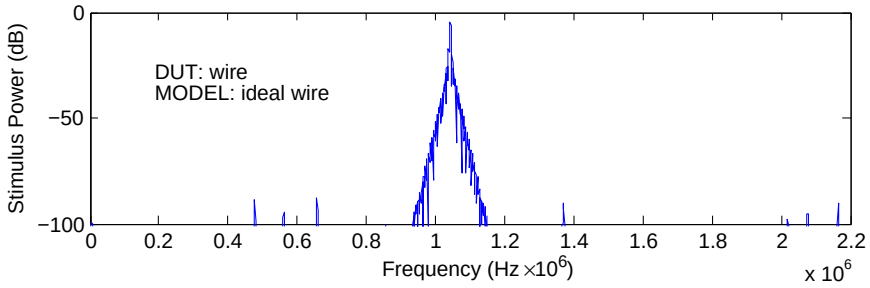

Figure 2.5: Best Stimulus from Experiment 2.1.2(a)

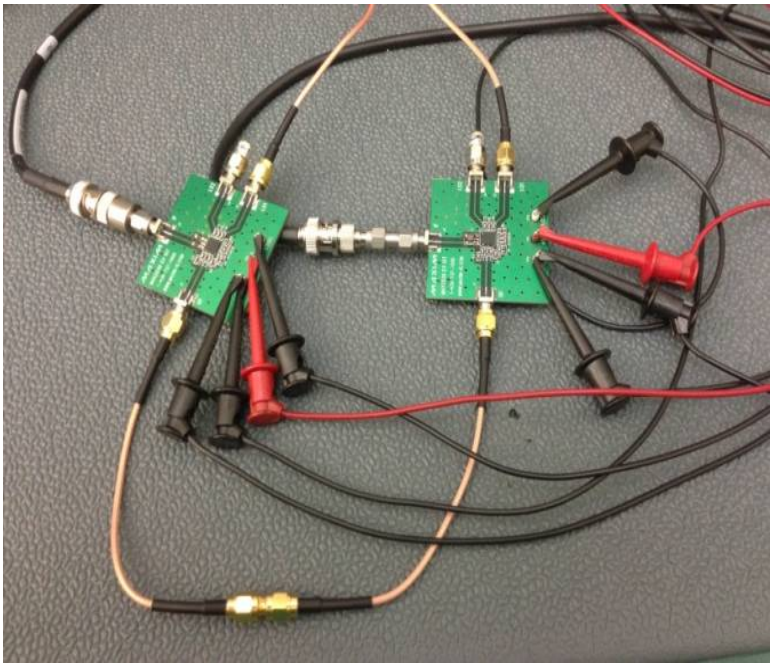

Figure 2.6: Experiment 2.1.2 DUT

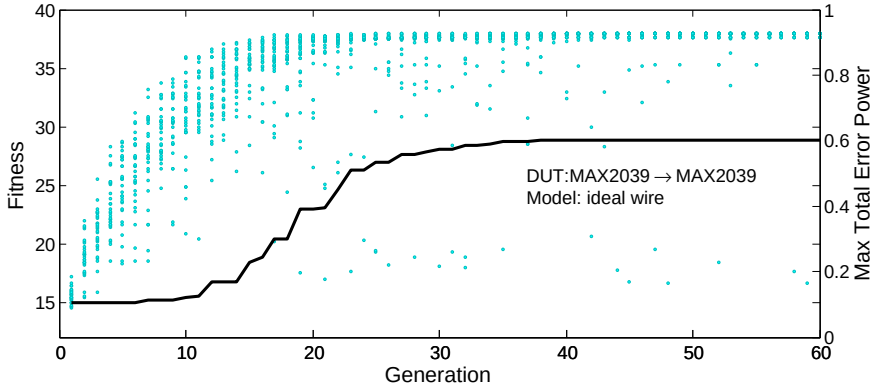

Figure 2.7: Experiment 2.1.2 Fitness Convergence

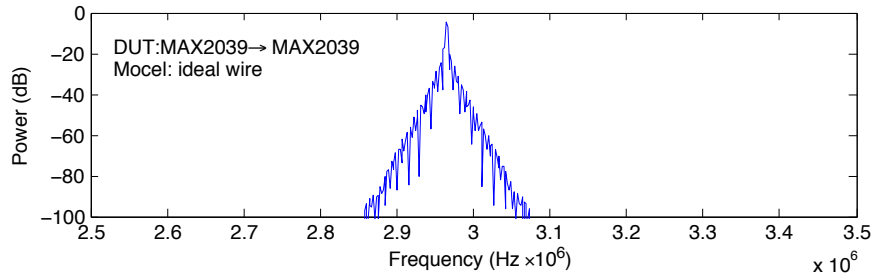

Figure 2.8: Best Stimulus from 2.1.2 (Frequency-Domain)

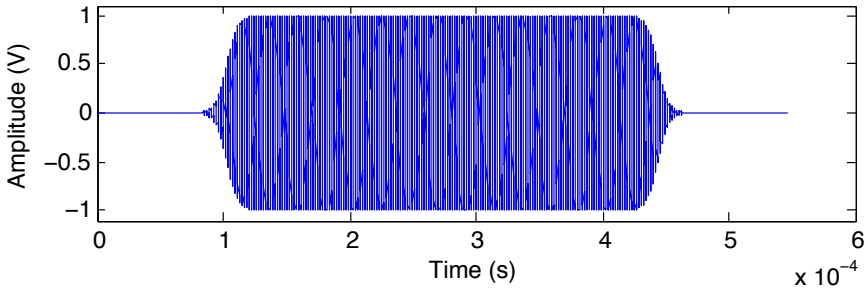

Figure 2.9: Best Stimulus from 2.1.2 (Time-Domain)

DUT=up/down mixers , MODEL=up/down mixers : This experiment was designed to test the effectiveness of iterative model solving throughout the RAVAGE process and to evaluate the ability of the non-linear solver to arrive at parameter values using a RAVAGE-produced stimulus as compared to [46]. As in *Experiment 2*, the DUT remains sequential up-conversion and down-conversion by a pair of MAX2039 converters with phase coherent LO inputs. In this experiment, however, a 5^{th} order polynomial model is used to model amplitude nonlinearities for each converter. The output of each module is given by:

$$M_{o_i}(x(t)) = \alpha_{0_i} + \alpha_{1_i}x(t) + \alpha_{2_i}x(t)^2 + \alpha_{3_i}x(t)^3 + \alpha_{4_i}x(t)^4 + \alpha_{5_i}x(t)^5 \tag{2.4}$$

Rather than use 0^{th} order parameters, a single DC offset parameter was used to capture otherwise unexplained DC on the output of the down-converter. The model was initialized to parameter values found by measuring one converter. The algorithm was stopped several times during the experiment to perform model fitting. Each time, a non-linear solver was employed to find model parameter values that would yield the least total error power. It should be noted that because the solver operates over the error PSD, it cannot distinguish between positive and negative α values (hereafter only absolute vales will be shown). During each call to the model fitting function, the top two stimuli were used in solving, and subsequent model fitting always included stimuli previously used in addition to those recently generated. Using a maximum stimulus bandwidth of 5 MHz, and an error BOI of 18

MHz, RAVAGE was allowed to run for 471 generations and model fitting was attempted 3 times. Fig.2.10 shows the progression of fitness and total error power as the model was updated with new parameters. *2.1.2* yielded 3 sets of calculated model parameters presented in table 2.1. The final model resulted in a maximum total error power of 3.841×10^{-3}, down from the initial maximum total error power of 7.892×10^{-3}. By the noise floor standard set in *Experiment 1*, all of these models exhibit total error powers far beneath those of *Experiment 2*. And finally, the performance specifications resulting from calculated parameters yield conversion-loss and IIP_3 figures consistent with the manufacturer specifications of $8.7\,dB$ and +$34.6\,dBm$ for the up-converter, and $9.49\,dB$ and +$35\,dBm$ for the down-converter; according to Maxim's datasheet, the acceptable range is: $7.1\,dB$ of conversion loss, a +$33.5\,dBm$ IIP_3 during up-conversion, and a +$34.5\,dBm$ IIP_3 during down-conversion.

2.1.3 Conclusion

The RAVAGE algorithm was successful in creating stimuli that expose unmodeled behaviors in the both the composite AWG/digitizer system, and the looped-back RF transceiver system. Though solution accuracy of model parameters cannot be guaranteed by RAVAGE's stimuli, we have shown that a better fitting model can be achieved. It's reasonable to believe that if parameter's values are restricted within the solver to practical and realizable values, any improvement can be accepted as plausibly accounting for some behavioral difference.

2.2 Exploring Limits of Parametric Soft-Fault Simultaneous-Sensitivity

In this section, I introduce a methodology for algorithmic generation of test signals for the detection of short and open-circuit defects in analog circuits [82]. Prior algorithms have focused on test generation for specific short- or open-defect values assumed at the start which places the burden of failure coverage on accurate analysis of observed defects in

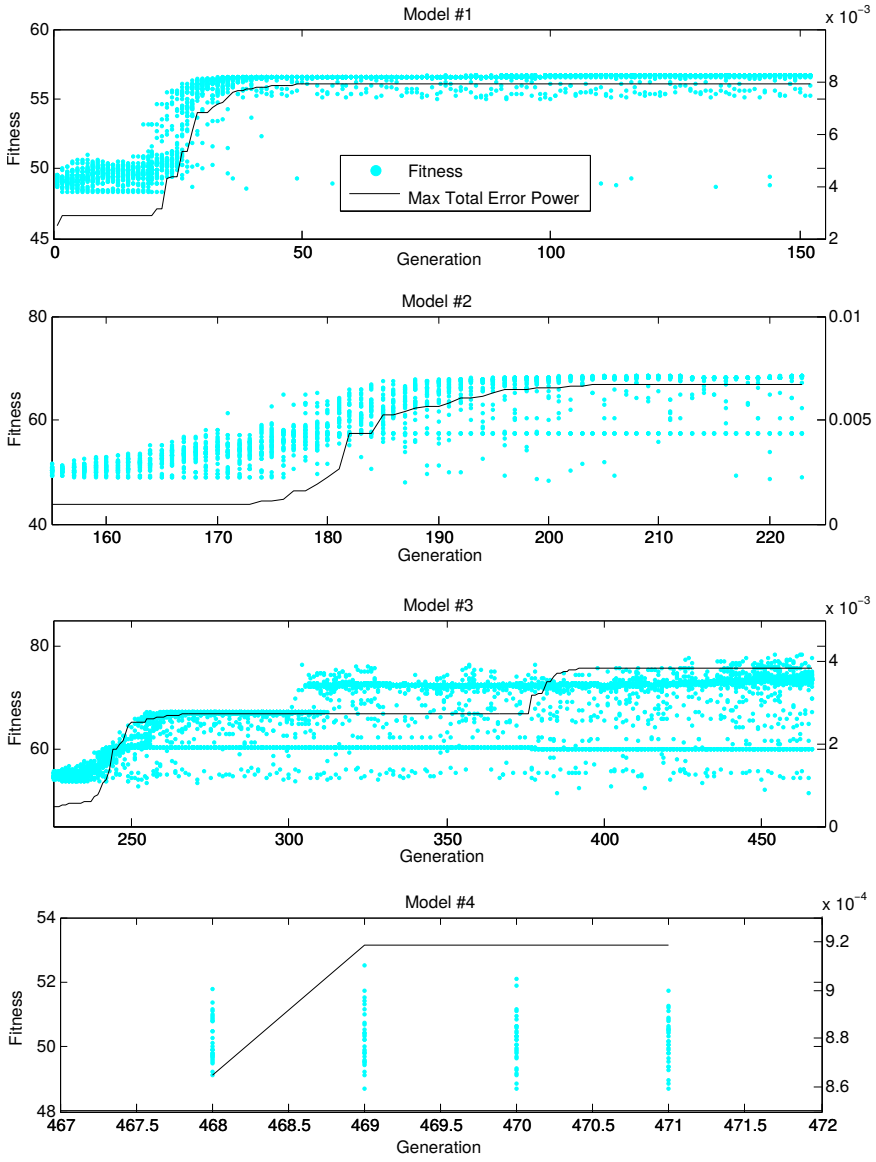

Figure 2.10: Experiment 2.1.2 Results

Table 2.1: Model Parameter Back-Calculation Results

	Parameter	Model 1	Model 2	Model 3	Model 4
Up-Converter	α_1	0.412	0.354	0.324	0.367
	α_2	1.54×10^{-2}	2.44×10^{-4}	6.77×10^{-3}	3.97×10^{-3}
	α_3	4.05×10^{-2}	1.05×10^{-3}	1.24×10^{-3}	1.69×10^{-3}
	α_4	0	1.95×10^{-4}	3.95×10^{-4}	4.81×10^{-4}
	α_5	0	4.87×10^{-5}	4.65×10^{-4}	3.78×10^{-4}
Down-Converter	α_1	0.412	0.354	0.293	0.335
	α_2	1.54×10^{-2}	7.82×10^{-4}	6.42×10^{-3}	3.84×10^{-3}
	α_3	4.05×10^{-2}	1.02×10^{-3}	9.97×10^{-4}	1.41×10^{-3}
	α_4	0	1.80×10^{-4}	9.60×10^{-4}	7.47×10^{-4}
	α_5	0	4.37×10^{-5}	2.22×10^{-4}	1.72×10^{-4}
	DC offset	0	$10.31mV$	$10.34mV$	$10.36mV$

known failed parts and comes at a high cost. In this work, we optimize the test stimulus to be simultaneously sensitive to detecting the *weakest* shorts and opens in analog circuits using a concurrent stimulus and defect value optimization algorithm. Since the defect value itself is an optimization parameter, the responses of nonlinear circuits corresponding to *multiple defect values* are considered as opposed to a *single* linearized representation corresponding to a fixed defect value as in the existing state of the art. The algorithm produces a test stimulus along with values of the weakest shorts and opens that the stimulus can detect (the locations of the defects are specified to the algorithm). These values are determined by the design of the analog circuit itself and therefore subsume all specified detectable defects for the circuits concerned. Experimental results show the feasibility of the proposed approach on selected test cases and defect sets.

The objective of this work is, given an analog/mixed-signal/RF circuit and list of internal nodes, to design a test according to the criteria:

1. The device is simultaneously sensitive to any specified short- and open-circuit faults.

2. All shorts and opens should be sensed in their least severe manifestations.

3. Utilize primary inputs, power supply voltage, and loading of the circuit under test to increase sensitivity.

4. Observe only the DUT's primary outputs.

The key advancement of this test generation algorithm over prior art is that existing test generation algorithms produce one test for one fault with a statically assigned magnitude, while our algorithm produces one test for many faults with dynamically determined minimal magnitudes. For example, given a short-circuit defect modeled by the presence of an extraneous capacitor with value C_{short}, the sensitivities of the outputs of the circuit containing C_{short} depend in general on the magnitude of C_{short}. Therefore, in the presence of nonlinearities, the sensitivity can only be maximized for a given C_{short}, which is not known *a priori*.) Existing test stimulus generation algorithms cannot address detection of defects across a large dynamic range of values of C_{short}, R_{short}, etc., due to linearization techniques that do not work well for devices with inherent nonlinearities (R_{short} being another example of a circuit element used to modeling a defect).

2.2.1 Algorithm

Statistical methods similar to those used in [24, 48, 58] are used to bring forward distinguishing subtleties in underlying circuit behaviors and leverage that information to predict other features of a circuit, in this case, the presence of a defect. In some cases, linear circuits can be made to operate in modes of reduced linearity in order to excite behaviors that are more sensitive to the presence of defects. We conduct such a search over the space of input signals while simulating the circuit under varying fault parameterizations. We draw attention to the fact that the circuit itself (i.e. faults/defects/process) is not a point, but it is a continuous probability distribution. Figure 2.11 illustrates an n-dimensional space of circuit outputs over varying stimulus, defect type, and defect magnitude. For large circuits, sampling techniques must be utilized in selecting defects to simulate [19, 31].

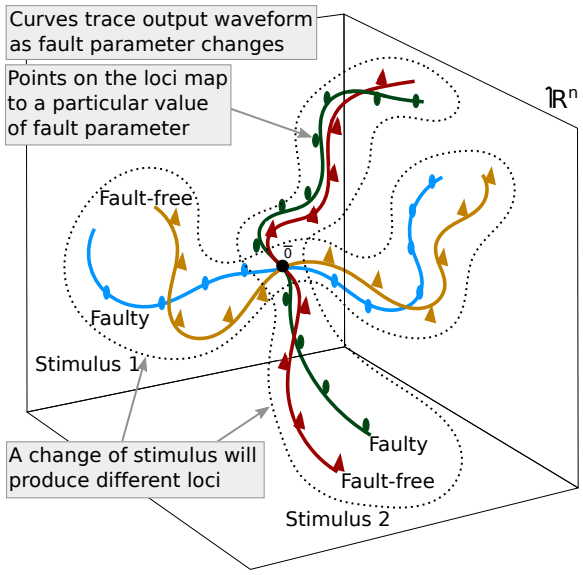

Figure 2.11: Stimulus Generation Problem Visualization

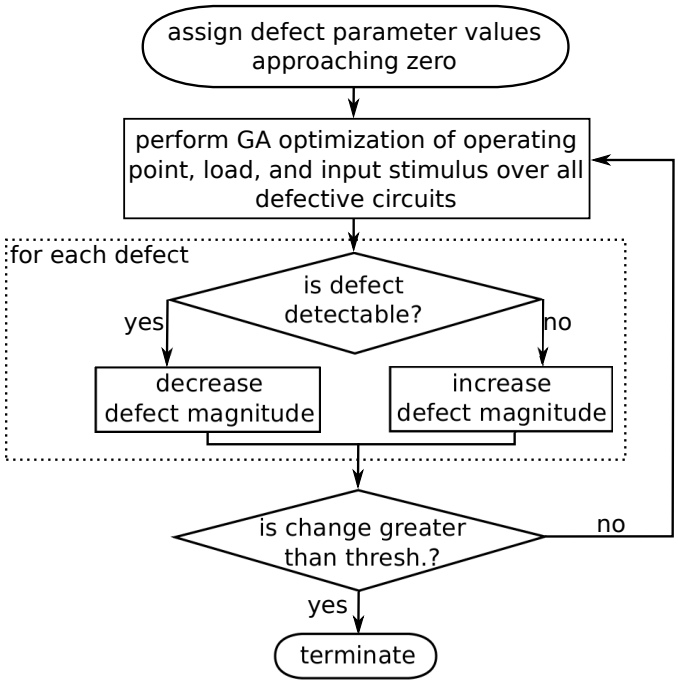

Figure 2.12: Test Generation Algorithm

In order to simulate the circuit, we must model the defect within the circuit. For example, a short-circuit defect can be modeled by the presence of a single additional resistor within the circuit; it is parameterized by the resistance value, with a large value corresponding to a "mild" short, and a small value corresponding to a "severe" short. Designing a test for a short circuit is specific to the chosen model and parameterization; in the presence of nonlinearities, *one cannot guarantee that the test holds for other parameterizations of the model without simulating them*. Figure 2.11 is an example visualization of response loci for two circuits, faulty and fault-free, simulated over a continuum of defect parameter values for two different stimuli. In order to detect a particularly parameterized defect, the corresponding point on its locus must be sufficiently distant from that of the defect-free circuit in the output space (where dimension n corresponds to sample n of the output waveform). In order to diagnose a defect, the corresponding point must not only be distant from that of the defect-free circuit, but it must also be sufficiently far from the loci of all parameterizations of all other defects considered.

In choosing a stimulus for defect detection and defect diagnostic tests, our optimization favors those for which pairwise response distances are maximal over all defect parameterizations. That is, stimuli are optimized such that the points representing each device's (defect-free and defective) responses are vertices enclosing a maximal volume. Note that some optimization techniques can be confounded by the high-degree of nonlinearity of the volume metric (Equation 2.5); for this reason, and due to the underlying nonlinearities exploited, the genetic algorithm (GA) is a suitable choice for optimization.

$$\text{n-dimensional volume} = \prod_{i,j} \text{dist}(x_i, x_j) \tag{2.5}$$

Though large volumetric distance eases the tasks of both detection and diagnosis, it can be difficult to achieve and is not necessary; adequate distance in only one dimension is sufficient for distinguishing vectors. For this reason, the pairwise distance, $dist(x_1, x_2)$

is not taken as the Euclidean norm, but rather higher order norms (Minkowski distance). Individual samples of greater separation are better representations of response distance. In the presented experiments, $p = 4$.

At this point, it's worth noting that various statistical methods (PCA, SVD, etc.) have been used to reduce the dimensionality of output waveforms, and metrics in those spaces have been used as a distance metric [34, 41, 42]. This technique can be perilous for at least two reasons: many of those methods operate on centered and normalized data, so unless overall magnitudes are explicitly included, the inter-signal variances might not be large enough to measure; and secondly as mentioned earlier, sufficient distance in only one dimension is adequate.

Because each point in the output space requires one simulation, and many evaluations are required throughout the course of optimization, there is a clear need for reducing the dimensionality of the problem in some way. Many have proposed means of accomplishing reduction: linearization of different variables, considering a limited set of defects (dictionaries), some have assumed quadratic behavior of cost function, some have proposed random sampling, etc. It is clear that if anything useful is to result, the reduction of the dimensionality of the problem must be done very carefully to retain effectiveness and to produce a test which can be used in application. If done without care, one might face months of simulation time, might produce tests which cannot be seen beneath the instrumentation noise of the test equipment, or may find their test ineffective across the distributions of defects and circuits encountered in real life.

The major contribution of this work is in providing a methodology for generating a test for detecting the presence of any single defect, parameterized to a minimally detectable state, from a set of parametric defects. One might assume that if the minimally invasive case is detectable, then defects with more egregious parameter values would also be detectable. In Section Section 2.2.1 we describe the algorithm; in Section 2.2.2, we examine an experimental test-case on a Low Noise Amplifier circuit; and in section 2.2.2,

we examine an experimental test-case on a low-speed elliptic filter.

The proposed algorithm involves two loops. The inner loop for stimulus optimization is formulated as follows: Given a particular set of circuits including defective and defect-free, perform a search over operating conditions (supply, loading, etc.) and input stimuli simultaneously, such that the outputs of all circuits are maximally separable in as many dimensions as possible according to the fitness function described in Equation 2.6. The outer loop determines whether or not the differences among individual circuits' responses are detectable considering the noise floor of the circuit and test equipment. For each defect instance, the minimum detectable value and the maximum undetectable magnitude of the defect is stored. The outer loop then derives a new set of defective instances with parametric values informed by the knowledge of the history of detectable and undetectable values, and re-enters the first loop (binary search). Upon termination, the last two tests applied are concatenated to yield a test for which all defects are detectable and the magnitudes of each defect's parameters are on the edge of detectability.

The genetic algorithm requires a scalar to represent the fitness of the population. We began with a Euclidean distance metric, but realized we required bias towards the largest pairwise distance to reflect our criterion for detection (sufficient difference between any single pair of samples). That is, a better measure of the fitness of the stimulus in this context will be more sensitive to the dimension of maximal separation than to the mean separation over all dimensions (in this work $q = 4$):

$$\text{fitness} = |\hat{x_i} - \hat{x_j}|_q \qquad (q > 4) \tag{2.6}$$

The search space is quantized and bounded to limit optimization time. A single quantization level is chosen for all optimization dimensions. For large circuits, the number of interior nodes grows at least polynomially, and so

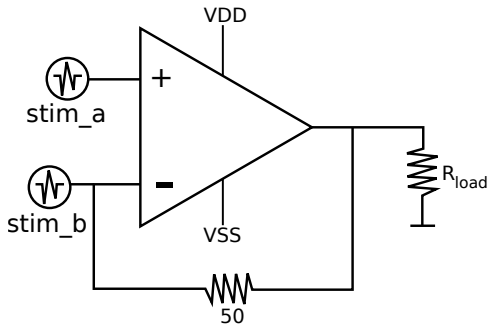

Figure 2.13: Op-Amp test setup.

2.2.2 Completed Experiments

Operational Amplifier: In this experiment, a 100 MHz bipolar opamp spice model is used as the test vehicle. Twenty faults were chosen at random from a complete set of node-to-node short-circuit faults and open-circuit faults at each terminal of each device. The comprehensive set totaled 406 possible unique short and open faults. The opamp was configured with a single 50Ω resistor in the negative feedback loop (Figure 3). Input stimuli were provisioned at the inverting and non-inverting inputs of the amplifier, and an output load resistor of variable size was installed. The algorithm was allowed to optimize VDD values, VSS values, load resistor values, and stimulus waveforms while comparing the responses of the 20 faulty circuits under varying fault magnitudes. The resulting testbed parameters are:$VDD = 5.76V$, $VSS = -6.43V$, $Rload = 4.49k\Omega$. Figure 2.14 shows the superimposed output waveforms of all faulty circuits at their minimal detectable magnitudes, and Table 2.2 shows the minimal detectable values. Figure 2.15 shows the two stimulus signals used.

Table 2.2: Minimally Detectable Opamp Fault Magnitudes

Fault Name	Testable ?	Fault Magnitude	Fault Name	Testable ?	Fault Magnitude
open 377	yes	154 Ω	short 258	no	-
open 333	no	-	short 135	no	-
open 356	yes	154 Ω	short 317	yes	80.6 MΩ
short 95	yes	80.6 MΩ	short 102	no	-
short 156	yes	34 MΩ	short 134	no	-
short 6	yes	80.6 MΩ	short 151	yes	80.6 M Ω
short 293	yes	65 MΩ	short 189	yes	80.6 MΩ
short 84	no	-	short 261	yes	80.6 MΩ
short 190	yes	80.6 MΩ	short 124	yes	80.6 MΩ
short 195	yes	80.6 MΩ			

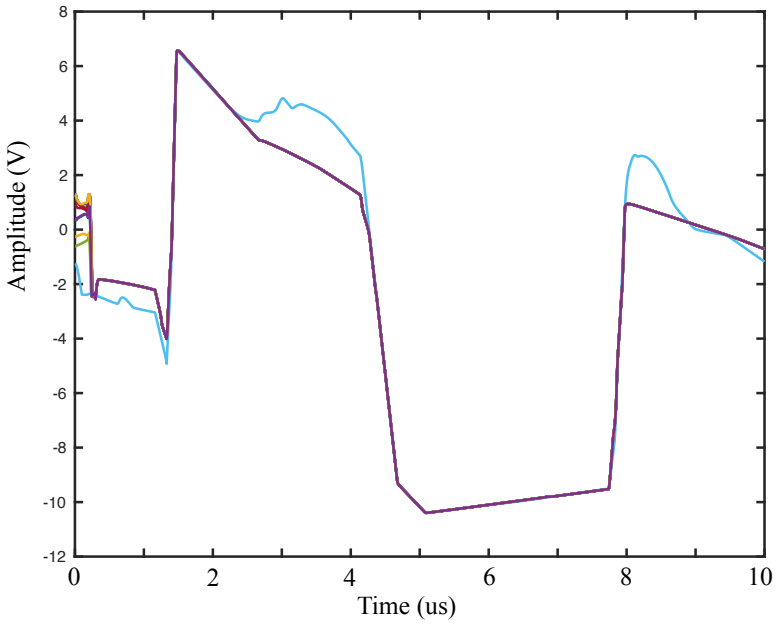

Figure 2.14: Responses of twenty marginally detectable faulty opamps, Experiment 2.2.2(z).

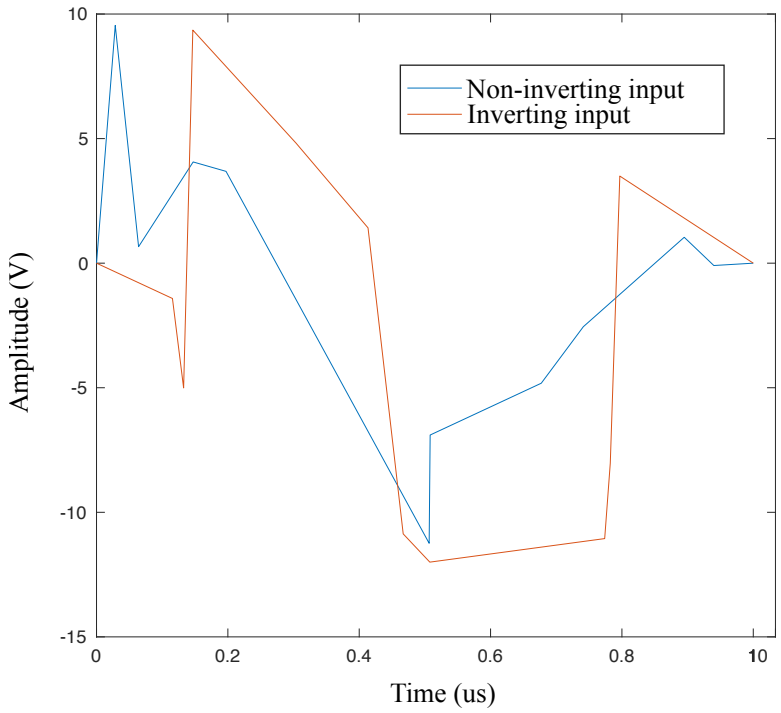

Figure 2.15: Test stimuli for opamp circuit, Experiment 2.2.2(a).

Low Noise Amplifier: In this experiment, a spice model of a 2.0 GHz low-noise amplifier (LNA) in 180nm CMOS is used as a test-vehicle. The design was first functionally verified in the defect free case, then seven copies of its netlist were created. In each copy, an additional capacitor was introduced between an internal node and ground and its value was left parameterizable. The design operates on a nominal 3v supply, so bounds of [0 , 4] were used for *vdd, vbias1* and *vbias2*. An ideal quadrature modulator was used to create an RF signal from in-phase and quadrature baseband signals. An envelope following (”shooting”) analysis was performed and the output envelope was measured for all defect instances. The baseband bandwidth was limited to the range (500kHz, 2 MHz) by choosing a sample rate of 4 MHz and a maximum simulation time $t_{max} = 2$us. The stimulus was a piecewise-linear waveform with 8 vertices, each vertex taking an amplitude in the range [0, 0.5] volts and a time in the range [0, 2 us]. The output load was allowed to vary in the range [1, 1k] ohms. Each of these dimensions were discretized to 64 levels for the GA. The minimum detectable capacitances through each iteration of the algorithm's outer-loop is shown in

Table 2.3: Summary of LNA Experiment Parameters

Parameter	lower bound	upper bound	quant. levels
vdd	0 v	4.0 v	64
$vbias1$	0 v	4.0 v	64
$vbias2$	0 v	4.0 v	64
vin (amplitude)	0 v	0.5 v	64
vin (time)	0 s	2.0 us	64
vin (freq)	500 kHz	2 MHz	interp. by spice
R_{load}	1 Ω	1 kΩ	64

Table 2.4: Summary of LNA Experiment Results

Result	Value
vdd	2.98 v
$vbias1$	0.32 v
$vbias2$	2.86 v
R_{load}	667Ω
$C_{mindet.}$	{167, 31, n/a, 53, 53, 167, 32} fF

Figure 2.17; the optimal stimulus and collection of responses are shown in Figure 2.18. Results are summarized in Table 2.4.

As described in Equation 2.6, the GA effectively maximizes the minimum-dimensional-distances between defect responses and the defect-free response as calculated by Equation 2.6. Table 2.3 summarizes the parameters of the experiment.

Elliptic Filter: In this experiment, an elliptic LPF from the ITC '97 benchmark circuits was used as a test-vehicle. Given the particularly low frequency operation of this circuit, it was thought to test applicability of the algorithm to circuits with large time-constants. The design was first functionally verified for the defect free case [83], then fourteen copies of its netlist were created. In each copy, an additional capacitor was introduced between an internal node and ground and its value was left parameterizable. The design operates on a a nominal 12v supply, so bounds of [0 , 12] and [0, -12] were used for *vdd* and *vss* respectively. The single input was directly driven from a 50Ω source. A standard transient analysis was performed and the filter output was measured for all defect instances. The baseband bandwidth was limited to the range (500 Hz, 100 kHz) by choosing a sample rate

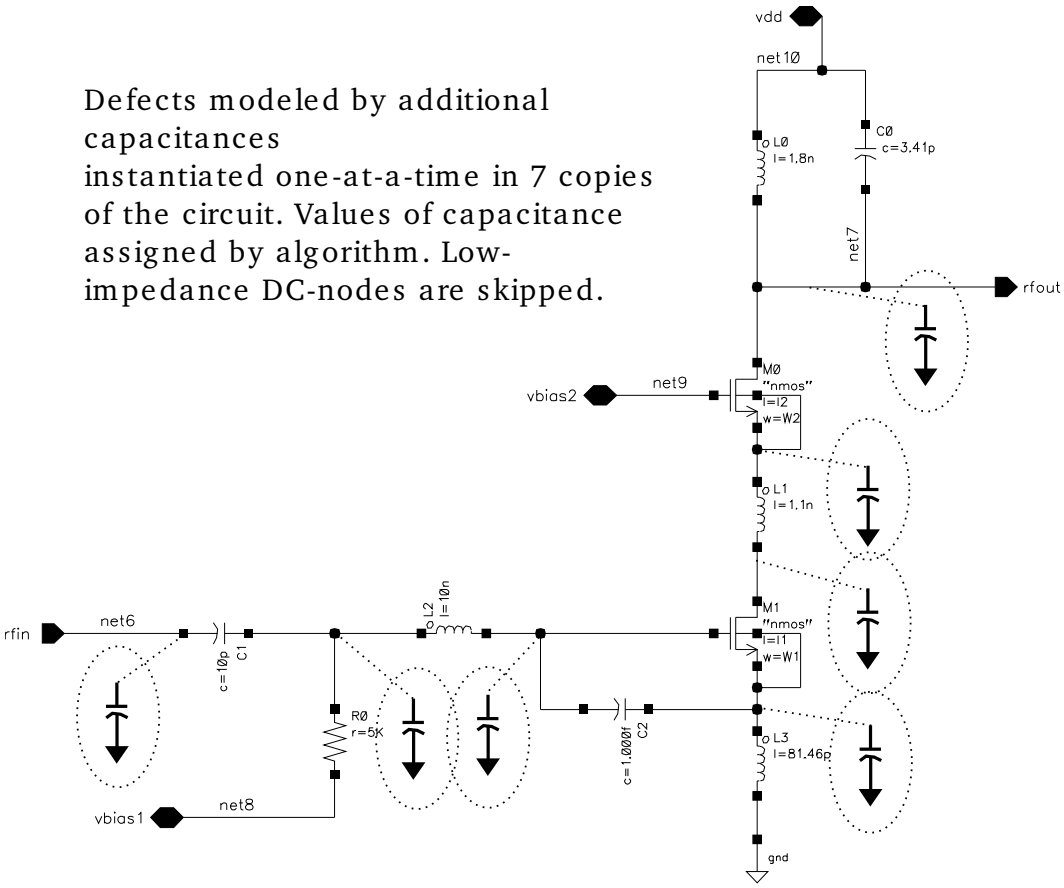

Figure 2.16: Low Noise Amplifier with Modeled Defects

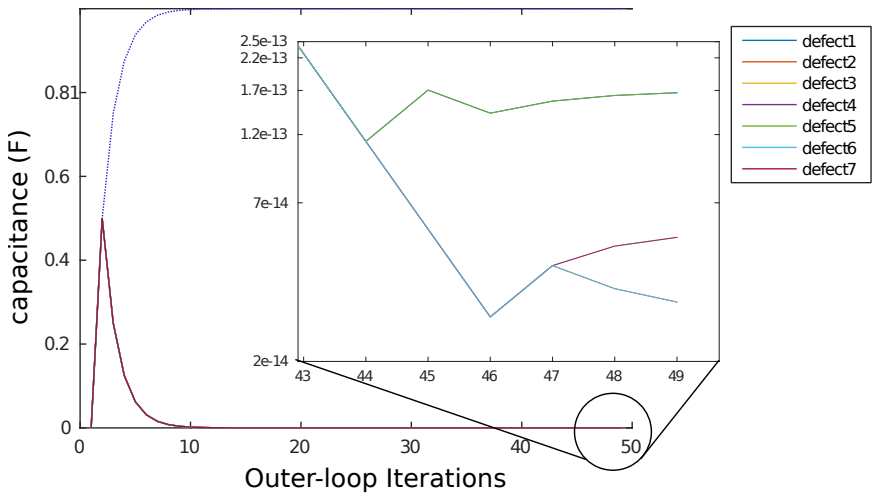

Figure 2.17: Evolution of Minimum Detectable Capacitance in LNA, Experiment 2.2.2(b).

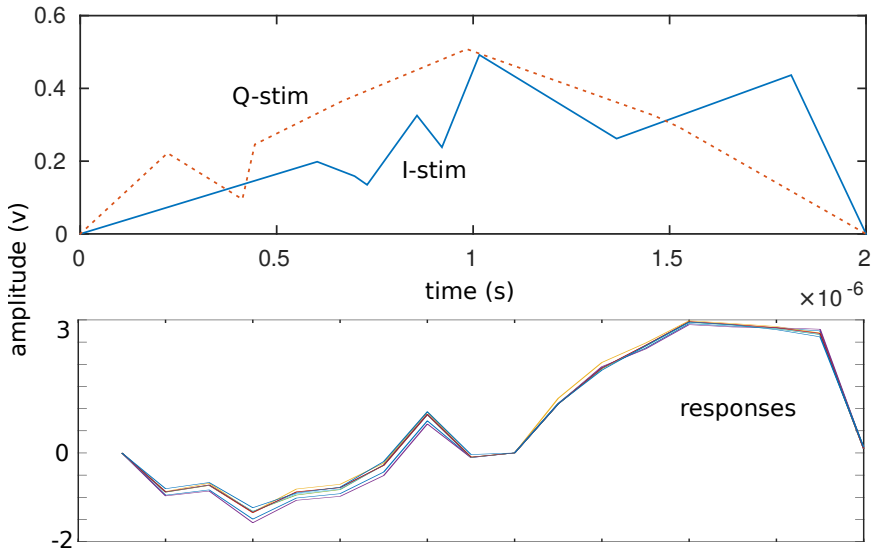

Figure 2.18: Baseband Stimuli (and Responses) which Achieved Min. Detectable Values in Experiment 2.2.2(b)

Table 2.5: Summary of Elliptical Filter Experiment Parameters

Parameter	**lower bound**	**upper bound**	**quant. levels**
vdd	0 v	12 v	64
vss	-12 v	0 v	64
vin (amplitude)	-10 v	10 v	64
vin (time)	0 s	2.0 ms	64
vin (freq)	500 Hz	100 kHz	interp. by spice
R_{load}	1 Ω	10 kΩ	64

of 200 kHz and a maximum simulation time $t_{max} = 2$ms. The stimulus was a piecewise-linear waveform with 16 vertices, each taking an amplitude in the range [-10,10] volts, and time [0, 2] milliseconds; the output load was allowed to vary in the range [1, 10k] ohms. Each of these dimensions were discretized to 64 levels for the GA. As described in Equation 2.6, the GA effectively maximizes the minimum-dimensional-distances between defect responses and the defect-free response as calculated by Equation 2.6. The minimum detectable capacitances through each iteration of the algorithm's outer-loop are shown in Figure 2.20. Results are summarized in Table 2.6.

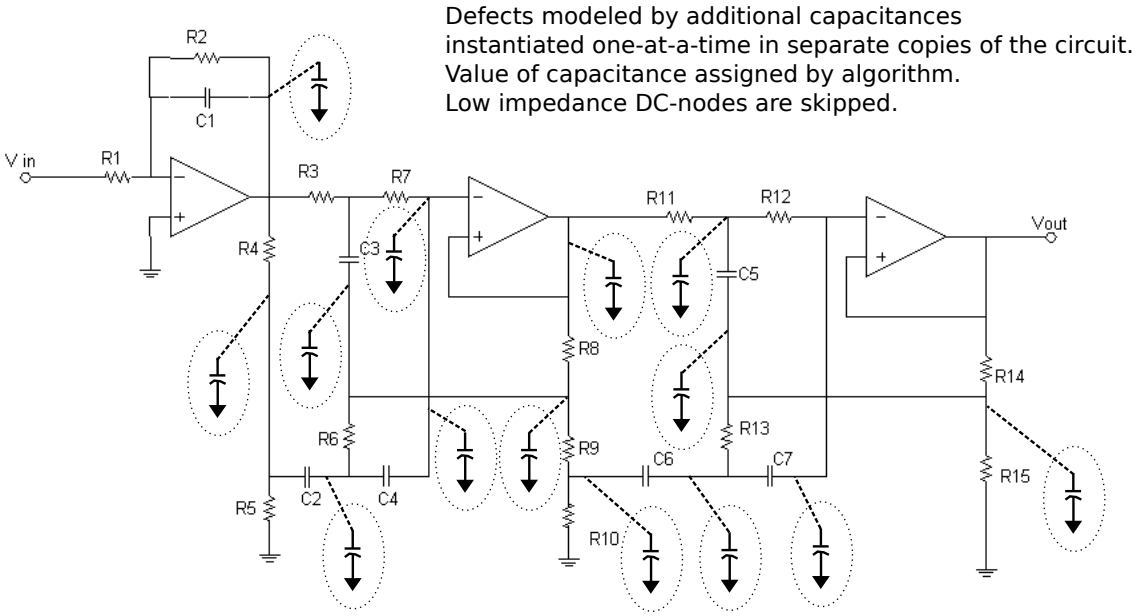

Figure 2.19: Elliptic Filter with Modeled Defects [83]

Table 2.6: Summary of Elliptic Filter Experiment Results

Result	**Value**
vdd	6.09 v
vss	-9.33 v
R_{load}	4.6 kΩ
$C_{mindet.}$	{155p, 1.7n, 1.7n, 4.9u, 667p, 784p, 253p 100n, 509n, 0.9, 252p, 667p, 369n,0.92} F

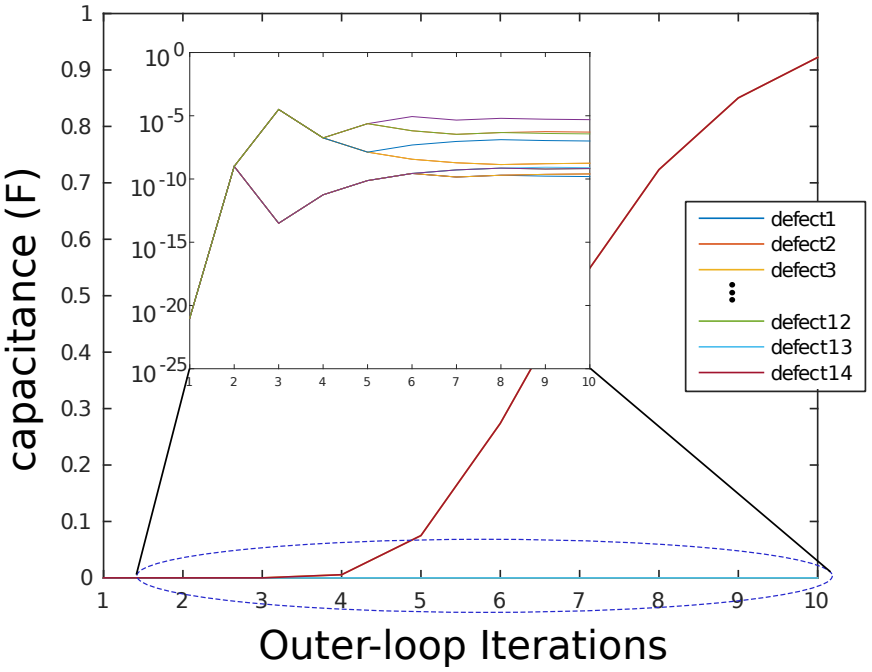

Figure 2.20: Evolution of Minimum Detectable Capacitance in Elliptical Filter, Experiment 2.2.2(c)

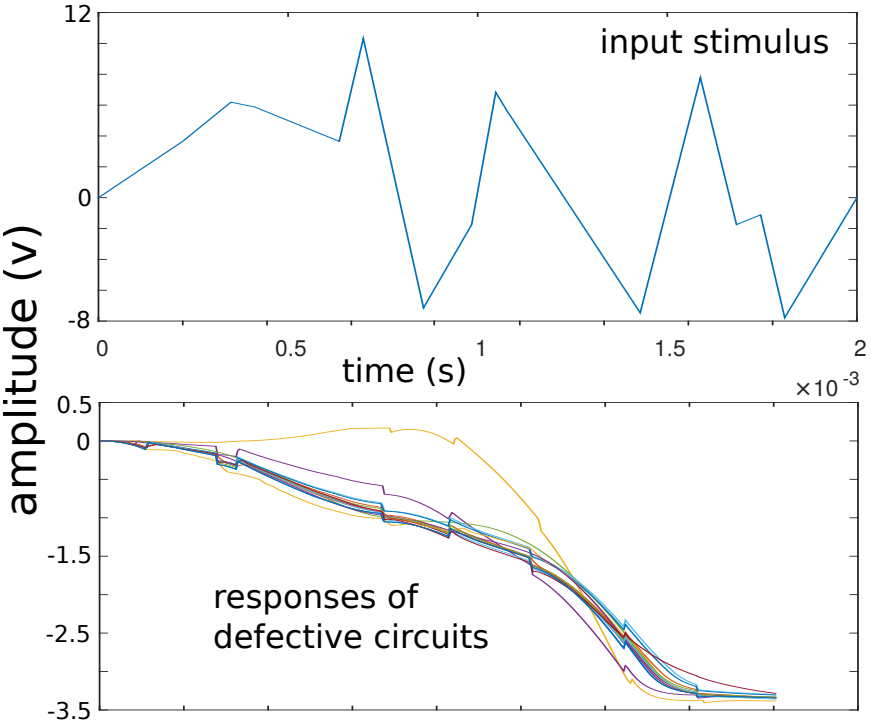

Figure 2.21: Stimulus (and Responses) which Achieved Min. Detectable Values for Elliptic Filter, Experiment 2.2.2(c)

2.2.3 Conclusion

Some capacitive shorts to ground were undetectable, most likely because of the low (zero) source impedance found at supply nodes of the simulation model. Other tests were sensitive to very small capacitances relative to the design values, in some cases down to the tens of femtofarad. The results were carefully reviewed and it's likely attributed to absence of device noise in the simulation. Given the magnitude of the design capacitances (2.7 nF) in the elliptic filter, it was expected that the minimum detectable values be found to be no smaller than 1 nF or so, yet the expectation was surpassed.Additionally, Looking at values of supply and bias voltages at which the solver converged, It is apparent that a state of reduced linearity has aided in the successful detection of the capacitive defects, perhaps hinting that a degree of harmonic generation and intermodulaion in the mosfets is aiding in sensitizing nodes to small capacitive loads. Additionally, investigation of the maximally advantageous way to apply this technique in diagnosis must be explored. In these simulations, a binary search was conducted from a minimum-capacitance initial condition. While the searches did converge, they did so in leaps and bounds, and thus future work will look into means of ensuring some way of quantifying certainties in defect magnitudes between sample points. Including device noise could have provided a better understanding of how realistic these measurements might be, however it would come at added cost in simulation time. Finally, a look at the effectiveness of various classifiers in distinguishing among differently loaded nodes and nodes loaded with capacitances of large variation will be explored.

2.3 Behavior-Learning and Modeling of Nonlinear RF-PA

In this section, I present a novel tool for post-silicon validation of mixed-signal/RF circuits through cooperative test stimulus generation and behavior learning. The implemented technique leverages iterative supervised learning techniques to comprehensively diagnose anomalies between the input-output behavior of the silicon device and the corresponding

behavior predicted by its reference model, resulting in both a more efficient validation test stimulus and an improved behavioral model which captures non-ideal silicon behaviors. Preliminary results on RF devices prove the feasibility of the proposed validation methodology.

2.3.1 Algorithms

As mentioned in Section Section 1.7 , post-silicon validation of mixed-signal/RF circuits presents the key challenges:

1. What test stimuli should be applied to the DUV in order to expose behavioral differences between the DUV's model and its implementation in silicon?

2. How can the behavioral model of the DUV be modified or augmented to best capture behavioral discrepancies between the DUV and its reference model?

In general, pre-silicon validation techniques fail as a result of modeling and/or simulation shortcomings arising from unknown physical interactions and any tests derived from these models can also fail [21]. Some discrepancies, those due to parametric differences, can be accounted for through parametric tuning [24, 36, 46, 48]. Post-silicon validation however, seeks to expose anomalies *beyond* parametric variation.

Stimulus Generation

Stimulus generation is performed using a revision of the RAVAGE tool. Similarly to RAVAGE, test generation is performed in an iterative manner over several passes; in each pass, a dominant behavioral discrepancy between the DUV and its model is excited, the discrepancy is then integrated into the model, and test generation is performed again to reveal other behaviors potentially masked in earlier iterations. The key contribution of this work is that behavior learning extends beyond model parameter tuning through the incorporation of artificial neural networks (ANNs) into the pre-silicon model.

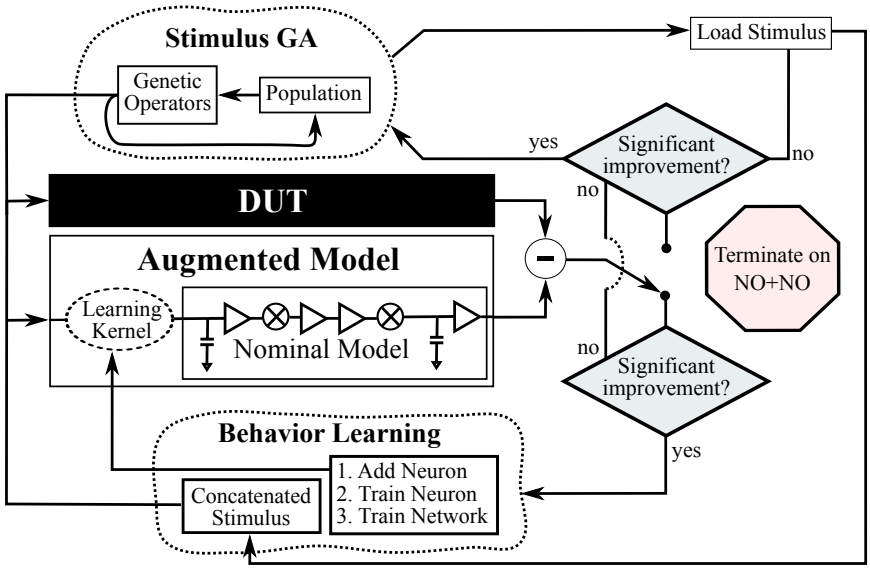

Figure 2.22: Embedded Learning - System Overview

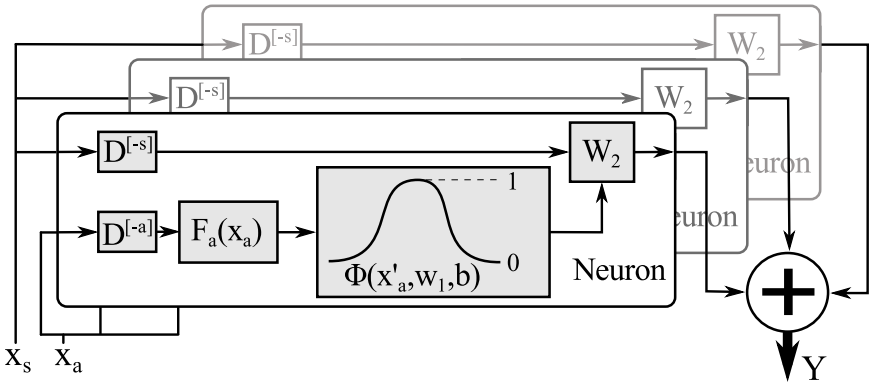

Figure 2.23: A Three Neuron SWN

Model Augmentation

Behavioral models are created and parameterized based on expectations, and in general, their functionality can vary across at least as many dimensions as there are parameters. However, one can easily imagine a case in which a silicon DUV exhibits behaviors which its model lacks: no set of parameter values adequately describe its behavior. It is for this reason that a highly flexible, highly adaptable, and easily trainable behavior-learning element is required to provide a means of selectively imparting additional degrees of freedom to parametric behavioral models. We began our work using low-order radial-basis networks (RBN), but as behavior complexity increased, we have come to use a variant of the RBN, described in Section 2.3.1.

The Sparse Wiener Network as a Learning Kernel

The Sparse Weiner Network is an artificial neural network which models a Weiner filter and which implements a sparse number of delay taps. Individual neurons in the SWN are structurally similar to the neurons in Radial-Basis Networks (RBN) , with the major distinction that SWN neurons allow for neuron activation independent of the input whereas RBN neurons are activated by and process the same input. This allows for the activation of neurons to originate in other signals and to be of a dimension less than or equal to that of the signal input. Figure 2.23 depicts an individual neuron inside an SWN.

Training SWN Learning Kernels within Behavioral Models

In order for a behavioral model to leverage the adaptability of the SWN, the two must be integrated into an augmented model. Because SWNs are used to learn the behavior of silicon DUVs whose internal signals are not observable, the SWN's target output is not known; therefore traditional training algorithms cannot be used. We perform a nonlinear optimization on neuron parameters during forward simulation only. It is assumed that if a network's neurons are added and trained individually to some locally optimal state, then

Algorithm 1 Algorithm for SWN Training

Input: DUV, $model$, $testStim$
Output: w_1,w_2, and b vectors

$modelout$=sim($model$,$testStim$)
$duvout$=sim(DUV,$testStim$)
$error$=mag($modelout - duvout$)
for n=1:$maxNumNeurons$ **do**
  $w_{1,n}, w_{2,n}, b_n$ = addNeuron()
  ROI=find($error$==max($error_vec$))
  for i = len($actFnList$) **do**
    $actin$=fwdTrace($model$,$testStim$)
    $actout$=$actFun_i$($actin$)
    $w_{2,n,i}$=newtonMin($error$,$actout$(ROI),$w_{2,n,i}$)
    $b_{n,i}$ = newtonMin($error$,$actout$,$b_{n,i}$)
    $w_{1,n,i}, w_{2,n,i}, b_{n,i}$=
      newtonMin($error$,$actout$,$w_{1,n,i}, w_{2,n,i}, b_{n,i}$)
    $performance_i$ = sum($(modelout - duvout)^2$)
  end for
  implement(index($performance$==min($performance$)))
end for

the resulting network will be near a local optimum as well. This method does not guarantee a globally optimal network.

2.3.2 Experimental Results

RF Power Amplifier Validation: An RF transmitter system was set-up in hardware using Maxim Semiconductor MAX 2039 up- and down-converting mixers with a Maxim MAX2242 power amplifier (PA) in the RF portion of the chain. A system model was created using the manufacturer's supplied conversion gain for the mixers and the PA's nominal linear gain. Iterative stimulus generation was performed and error measurements were taken throughout (Fig.2.24). In total, 8 stimuli were found and 34 neurons were added and trained within the model. Figure 2.25 presents the pre-silicon assumption of behavior, the DUV behavior, and the augmented model's behavior. It is important to note the SWN's ability to capture the hysteretic behavior of the amplifier.

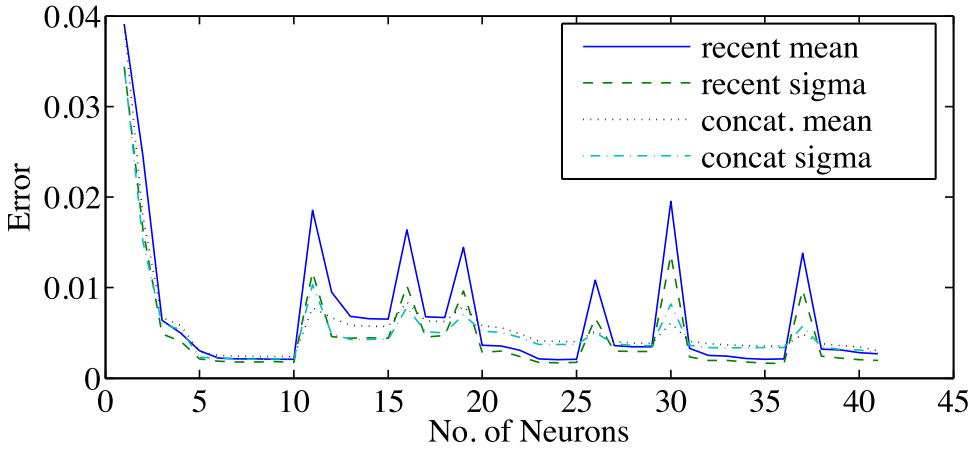

Figure 2.24: PA error over iterations, Experiment 2.3.2(a)

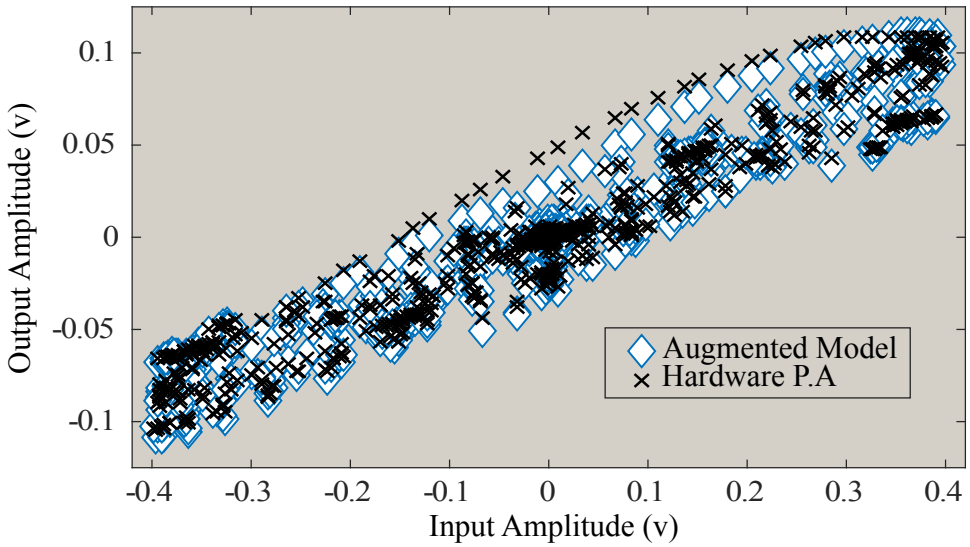

Figure 2.25: PA model performance, Experiment 2.3.2(a)

Full Transceiver Validation: A full transceiver chain was configured in "RF loopback," with the output of the transmitter externally attenuated and fed back into the LNA. Its model includes several nonidealities in each component of the chain. Both DUV and its model are Matlab simulation models, and the DUV is simply an instance of the nominal model whose PA has been modified to exhibit AM-PM effects through use of a 3rd order polynominal. It is noted that while the difference in description is limited to AM-PM effects within the PA, the systems' behaviors vary much more intricately than simple AM-PM behavior. The validation procedure was iterated 6 times and a total of 18 neurons were added. Fig. 2.26 depicts bare model performance across the entire stimulus space, while Fig. 2.27 depicts the augmented model's performance across the entire stimulus space.

2.3.3 Conclusion

In reducing both mean error and error sigma dramatically (Fig. 2.28 and Table 2.8), the sWN-augmented model was effective in incrementally learning the anomalous behaviors of the DUV as they were exposed by the RAVAGE algorithm. As thoroughly improved as the model became, a radial sliver of error of relatively large magnitude is seen in Fig. 2.27. This residual error seems to be an artifact of poor implementation of angular distance

Table 2.7: Nominal Model Parameters

Component	Param. Desc.	Param. Value	Unit
Upconv. Mixer	I-gain	0.985	v/v
	Q-gain	1.03	v/v
	I-phase $\delta\theta$	0.0524	rad
	Q-phase $\delta\theta$	-0.0698	rad
PA	linear gain	31.6	v/v
	3rd order	-59.598	v/v^3
	5th order	-53.7200	v/v^5
LNA	linear gain	5.6	v/v
	3rd order	-31.7901	v/v^3
	5th order	-17.6999	v/v^5
Dnconv. Mixer	I-gain	0.99	v/v
	Q-gain	0.975	v/v
	I-phase $\delta\theta$	0.02	rad
	Q-phase $\delta\theta$	0.03	rad

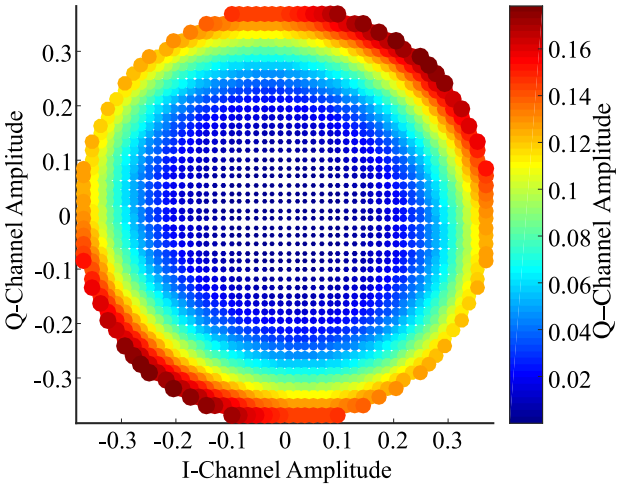

Figure 2.26: Uniform sampling of stimulus space, Experiment 2.3.2(b)

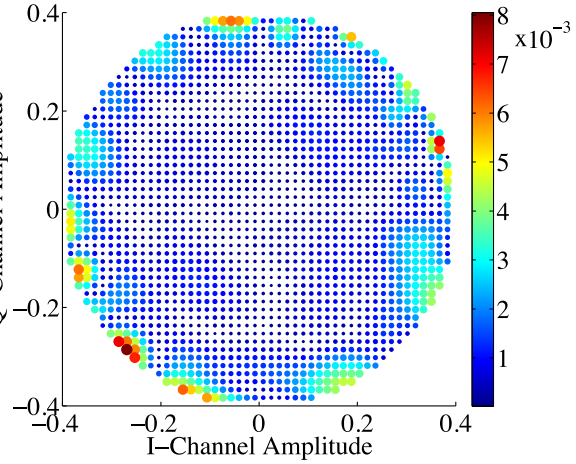

Figure 2.27: Uniform sampling of stimulus space, Experiment 2.3.2(b)

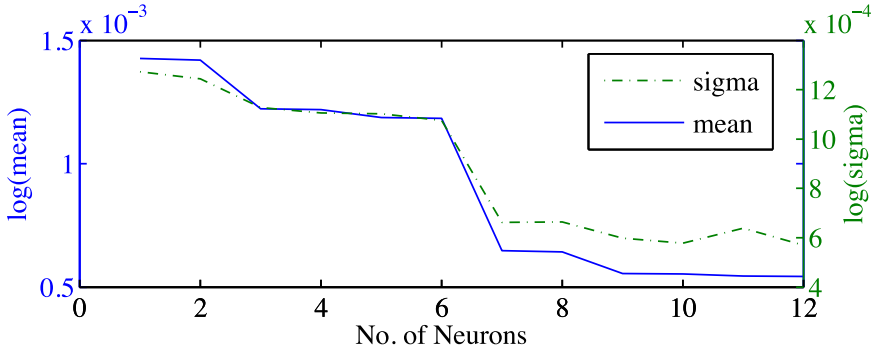

Figure 2.28: Mean error and error standard deviation as neurons are added to the SWN

Table 2.8: Selected Results

Feature	**Value**	**Unit**
nominal model mean error	20.7	mV
augmented model mean error	542	μV
nominal model error sigma	35.6	mV
augmented model error sigma	570	μV
neurons implemented	36	–
unique activation functions employed	7	–

measure in the SWN, and will be remedied immediately. Also, the question arises: at which point has the concatenated RAVAGE stimulus successfully stimulated all aberrant behaviors? Though the number of iterations through the algorithm presented in Sec. 2.3.1 was user-defined at 6 and SWN training had a termination condition based on improvement rate, perhaps better termination conditions could be found to balance the opposing goals of RAVAGE and the SWN training algorithm.

The test case validates the SWN as a tool for use in learning memoryless, 2-dimensional local scaling and rotational anomalies, and it also validates the training algorithm for use in cases where network output targets are not readily determined. Additionally, the training algorithm is shown to be effective when the network is heterogeneous, containing neurons of varying activation shapes.

Future work of this kind will include test cases to validate training algorithms to learn behaviors containing memory effects and to function within systems and components with dimensionality greater than two. Additionally, the authors will explore the use of the SWN in diagnosis of component faults within systems, and the ability to accurately characterize valid components when other components in the system are exhibiting anomalous behavior.

In this section we have presented a novel approach for automatically detecting and modeling design bugs in RF systems using test generation experiments on fabricated silicon. To the best of our knowledge, this is the first methodology of its kind that delivers design bug models to the designer and validation engineer for rapid design debug. The method has been validated through hardware and simulation experiments, and shown excellent results. We are currently applying the methodology to different classes of circuits and experimenting with different kinds of behavior learning algorithms to determine what gives the best results.

CHAPTER 3
REINFORCEMENT LEARNING APPROACHES

In this chapter, we reformulate the systems under validation as a Markov decision process and examine the use of reinforcement-learning to provide a globally convergent solution, a means of "storing" the valuable information created during stimulus generation, and low-cost iterated generation. The integration of the proposed design validation methodology with deep-Q learning software and the suite of Cadence simulation tools is presented, validation results for selected design bugs in representative designs are analyzed, and the quality and efficiency of the proposed design validation methodology is discussed.

3.1 Introduction

In hierarchical system design (SoCs, SoPs), models for mixed-signal components (e.g. regulators, data converters, I/O, RF) are described at the behavioral level (Simulink or MATLAB) with the complete system described as an interconnection of these models. The design strategy is typically top-down with bottom-up verification of synthesized netlists and physical layout. The design is incrementally advanced and verified with the intent of ensuring design correctness through each iteration of the design process from high level design specification to physical layout to silicon. This prevents expensive correction of design bugs that percolate from early stages of the design process to later steps including fabrication of silicon. To facilitate rapid turnaround design, it is necessary to verify behavioral equivalence between a higher level specification of the design (e.g. AHDL) and its lower level implementation (e.g. netlist) as early as possible.

While the analog specifications of modules at two different levels of the design can be checked for equivalence, such a check does not guarantee behavioral equivalence across the entire space of input stimuli because the set of specifications used to check for equivalence

may itself be *incomplete* [52, 84]. For similar reasons, formal methods for design equivalence checking may fail because designer inserted assertions may not cover all input-output behaviors and are generally myopic and constrained to specific input conditions.

3.2 Prior Work

The use of rapidly exploring random trees (RRT) for analog circuit test generation was presented in [85, 86]. The idea was to quickly explore reachable points in the state space of the analog circuit for verification purposes. In [87], an efficient discretized state space guided test stimulus generation approach is proposed with the goal of equivalence property checking. The methodology combines formal methods with circuit simulation techniques. In a similar vein, the equivalence between a behavioral model and its transistor level design over a highly likely input stimulus space is discussed in [88]. The discrepancy between the two design descriptions over the space of possible input stimuli is maximized to detect design errors. The work of [89–91] was among the first to combine formal verification methods with simulation driven techniques to explore the limits to which analog design specifications can be stressed, but assumes tests that are derived from manually crafted specifications. Central challenges going forward include eliminating up-front assumptions of input distribution (i.e. "kinds of input") and assumptions of completeness of any set of provided specifications.

3.3 Motivation

The purpose of this work is to present a new reinforcement learning (RL) driven test generation and anomaly-model generation algorithm that does not require any a-priori knowledge about: stimuli to be used for design verification/validation, the device specifications, or properties to be verified. Additionally, the RL techniques used have been shown to be convergent upon the global optimum under certain conditions and internally contain a useful distillation of all previous information (they have memory) and so can be reused and

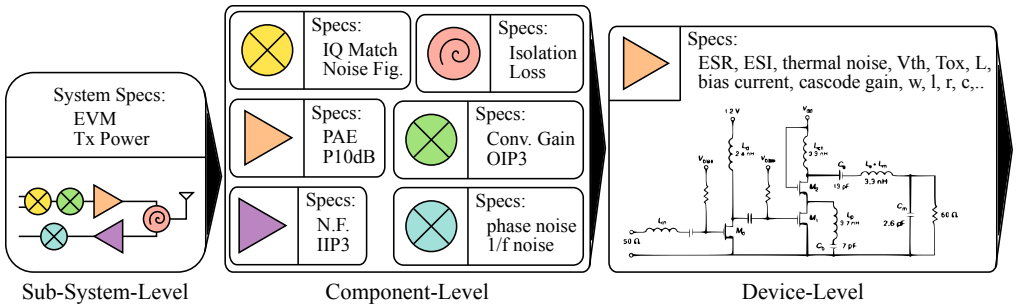

Figure 3.1: Illustration of a top-down design process

retrained many times and in many contexts at higher efficiency than optimization alone [92].

Traditionally, design requirements are defined by putting bounds on the values of many individual performance metrics and testing them explicitly. This method can clearly define what is valid and what is not; however, it is very difficult to completely constrain acceptable behavior of a system through performance specification, and complete sets of performance specifications are rarely available during design implementation. Though occasionally system-level performance may be calculable as an analytical function of subsystem parameters, this is not generally the case, and is confounded by components which exhibit memory effects, nonlinear behavior, or high-order physical phenomena. Contemporary system designers find themselves spending increasing quantities of resources on design debugging, functional verification, and post-silicon validation [1] and [93]. The authors of both [84] and [52] refer to corner cases and anomalies for which validation results are inconsistent with performance metrics:

1. Designs may meet existing performance specifications under certain tests, but fail in the anomalous case

2. Designs may not meet a particular performance specification, but only because the desired specification is unattainable in practice. Should such cases exist, it is the goal of design validation to identify them. Equivalently, if we can reject the claim that the system we've produced is equivalent to our intent, we wish to do so as quickly as

possible.

Explicitly, these tasks require the evaluation and comparison of the dynamical behaviors of two descriptions of the same system. If one can implement an ideal opamp using transistors, then any system containing opamps will be agnostic between the ideal model and the transistor implementation. Unfortunately, no transistor opamps are ideal. And so, as the descriptions of compositional blocks of systems become increasingly detailed, they become decreasingly ideal, and their behaviors stray further from their more abstract and idealized models.

This paradigm extends thorough fabrication, which can be thought of as the "most detailed" description of a system that can be achieved. Both validation and verification seek to first quantify the degree of behavioral departure from the architect's original intent and then classify it as either "valid" or "invalid," verification having been "passed" or "failed."

Design validation addresses the testing, evaluation, and comparison of the behaviors of the various subsystems and components expressed in different ways across the design hierarchy. Pre-silicon validation (conventionally "verification") involves evaluation and comparison of simulation models of systems, subsystems, and components at one description level against simulation models at other description levels. Post-silicon validation involves evaluation and comparison of the dynamical behaviors of simulation models of systems, subsystems, and components against the dynamical behaviors exhibited by physical systems. Both pre-silicon and post-silicon validation (pre-silicon being synonymous with design verification) seek to quantify the degree of behavioral departure from the designers' intent. It is the goal of AMS system validation to either support or reject the claim that a detailed system design (either physically or in simulation) meets the requirements of the system specification more broadly.

In Section 3.4 , we briefly review the subject of RL and its relationship to other machine learning (ML) techniques. In Section 3.5.1 , we present our formulation of the analog and mixed-signal (AMS) system validation problem as a Markov decision process (MDP).

In Section 3.3.1 , we illustrate the direct applicability of contemporary RL techniques to validation and explore the possible benefits over existing techniques. In Section 3.6 , we detail our implementation of deep Q-learning and discuss our experimental setups. We present experimental results and provide a summary analysis in Section 3.7 .

3.3.1 Validation as a RL Problem

Several approaches exist in the literature for generating time-varying signals (also "stimuli" or "tests") to be used for validation. These equate to identifying a *sequence* of inputs which, when applied to a circuit, will maximize the likelihood of observing behaviors that can lead to inference of a pass or fail. The authors in [85] utilize Rapidly Exploring Random Trees (RRT) which primarily emphasize coverage of the stimulus space as a means of evaluating candidate stimuli. The authors in [58] utilize genetic algorithms to identify chromosomal traits that lead to effective stimuli.

Reinforcement learning introduces to MDPs the notion of "reward." Reward is a measure of the desirability of a certain outcome. In reinforcement learning, it is the goal of the algorithms to manipulate the MDP via its inputs in such a way as to maximize the expected reward at the end of an experimentation period, or "episode." If we can devise a reward which aligns itself with our validation goals (to prove inequivalence, for example), then perhaps we can employ reinforcement learning techniques to find stimuli

3.4 Review of Machine- and Reinforcement-Learning

"Machine Learning" refers broadly to the study of statistical methods that leverage the power of computing hardware to predict attributes of future observations by making inferences from past observations, the "learner's" ability to do so is predicated on exposing the learner to a volume of historical data which is representative of future observations. The categories of tools that comprise machine learning are divided into two main classes: those that are "supervised" and those that are "unsupervised." A supervised approach refers to

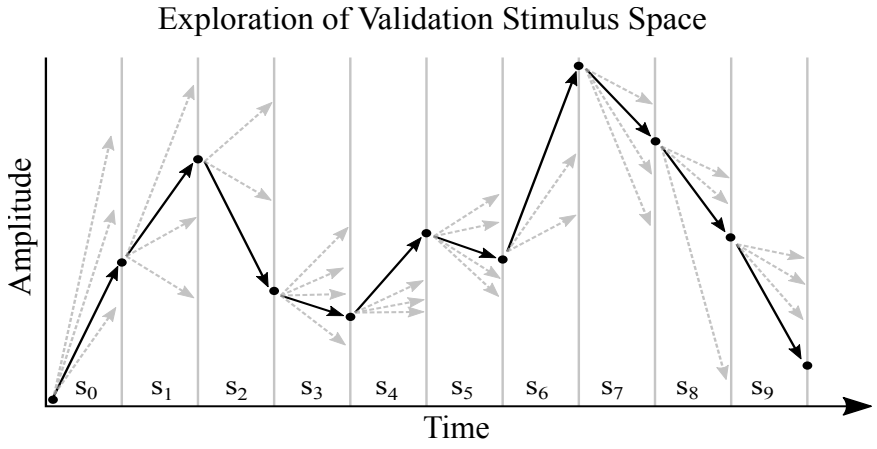

Figure 3.2: Validation stimulus built as a succession of actions taken, each leading the system into a different state, S_i

the requirement for human intervention in the preparation of training data. By contrast, "Unsupervised" algorithms do not require human intervention. Classically, unsupervised techniques consist mainly of data clustering algorithms.

3.4.1 Reinforcement Learning

In the early 1990s, work on a third class of machine learning algorithms called "reinforcement learning" began to circulate. [94]. In RL, rather than require human intervention on the very granular datum-to-datum scale, one provides a means for the machine to evaluate its own performance and guide its own learning. The full potential of RL became evident with the advancement of parallel computation tools, and the work of Minh et. al. ([95]) wherein a RL learner was trained to play the Atari video game console with skill surpassing that of humans put RL squarely into the spotlight. Google's DeepMind team has recently enraptured the public once again with their exhibition of a self-taught RL player which has handedly taken the crown from the reigning champion, Stockfish [96]. The overarching goal of RL is for a machine to learn to be "good" at a task in a self-directed fashion through feedback about its performance received via the reward signal (discussed in Section 3.3.1).

3.4.2 Basics of (Deep/Double) Q-Learning

Q-Learning is so named because it centers around the approximation of a "quality" function,

$$q_{i+1} = Q(s_i, a_i) \tag{3.1}$$

which predicts the "quality" resulting from taking a certain action, a_i, given that the system is in state s_i. It requires that every intermediate action, a_i, receive a corresponding reward signal, r_i. In this work, circuit state is defined by a vector of node voltages and branch currents. Individual actions are defined as ramp inputs to the system which take the inputs of the circuit under test from an initial value, a_i, to a_{i+1} as shown in Figure 3.2 . Starting from the state s_i, the RL takes an action (choice of input ramp) which will lead the system into state s_{i+1}. We also define Q^*, the discounted sum of all n incremental rewards received during an episode:

$$Q^*(a_0, \ldots, a_n) = \sum_{i=0}^{i=n} \sum_{i}^{n} \gamma^i r_i \tag{3.2}$$

where γ is the factor of time-discounting, which takes a value on the range [0,1] (and is usually close to 1).

The goal of Q-learning is to accurately predict Q at each step so that optimal actions are selected, resulting in maximization of the expectation of Q^*. In [92], the algorithm is shown to be convergent to a global optimum. In order to correctly model Q however, one must explore the entire state-space of the system which can be infeasible even for modestly sized digital systems. So instead, Q-learning algorithms balance exploration and exploitation in effort to deliver acceptable performance in reasonable time.

"Deep Q-learning" is simply an implementation of Q-learning which leverages a so-called "deep" neural net (one which has a number of layers in series performing sequential abstractions from observation space to inference space) to implement the Q function. Double-Q learning is another variation in which separate networks are used for action selection and Q estimation which can avoid systematic bias introduced by guiding learning

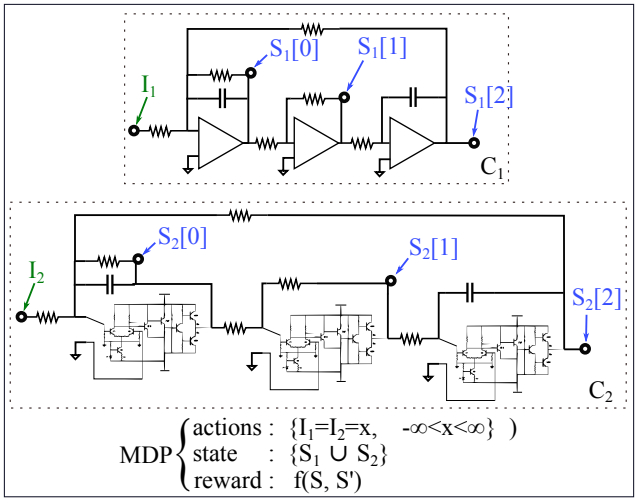

Figure 3.3: Example of two circuits combined to create a validation MDP

with the same network that is doing the learning.

In this work, we employed DQN algorithms as described by Minh, et. al. in [95] through open source implementations provided by OpenAI in their Baselines framework [97] as well as from the "stable-baselines" package [98].

3.5 Stimulus Generation for Behavior Discovery

3.5.1 Circuit-pairs as Markov Decision Processes

In this work we formulate the validation of a system as a Markov decision process (MDP): a state machine which transitions probabilistically as a function of input and state, with the transition probability functions being stationary (memoryless). As illustrated in Figure 3.3 , we compose such a system by exercising two systems in unison and observing their corresponding states. So state sets $\mathcal{S}_1$ and $\mathcal{S}_2$ must contain at least one node voltage or branch current which corresponds behaviorally (non-null intersection). The state of the MDP, s, is then the union of all pairs of states which $\mathcal{S}_1$ and $\mathcal{S}_2$ have in common, and the action inputs to our MDP map to a circuit input shared by the systems. Let m represent the number of *pairs* of states. Figure 3.2 illustrates the progression of a circuit through

time conceptualized as as sequence of state transitions resulting from a particular input sequence. In some cases where only a subset of circuit states are observable, previous values of input (up to the system's memory depth) can be used as a proxy for unobservable states. In such cases, a system with memory-depth N can be made to seem Markov to an observer if the previous N inputs are treated as states.

3.5.2 Validation and RL Reward

Reinforcement learning introduces to MDPs the notion of "reward." Reward is a measure of the favorability of an individual transition from one state to another. In reinforcement learning, the algorithm's goal is to exercise the MDP environment by carefully manipulating its inputs in a way that maximizes the expected accumulated reward at the end of an experimentation period, or "episode." If we can devise a reward which aligns itself with our validation goals (to prove inequivalence, for example), then we can employ reinforcement learning techniques to find stimuli which maximize the likelihood of reaching our goal.

Designing the Reward Function

In each time step of the learning episode we perform a short transient simulation composed of many timesteps of its own, and so every RL timesetp results in $2 \times m \times k$ data points. Choice of time discretization should be made such that observations appear **at most** weakly nonlinear within any RL time step. The reward function, therefore, must accept a $2 \times m \times k$ matrix as input and produce a scalar reward.

In validation, we are interested in the distance between the states of the two systems. There are many distance metrics which might be suitable in helping produce a scalar reward: L-2 norm, Manhattan, cosine, cross-correlation, Mahalanobis, Wasserstein, etc.; analysis of the relative merits of each is an area of active research and must be assessed on an individual basis [99]. In this work, we have chosen the L1 norm of the time-domain difference waveform as the reward metric for reinforcement learning. In experiment Sec-

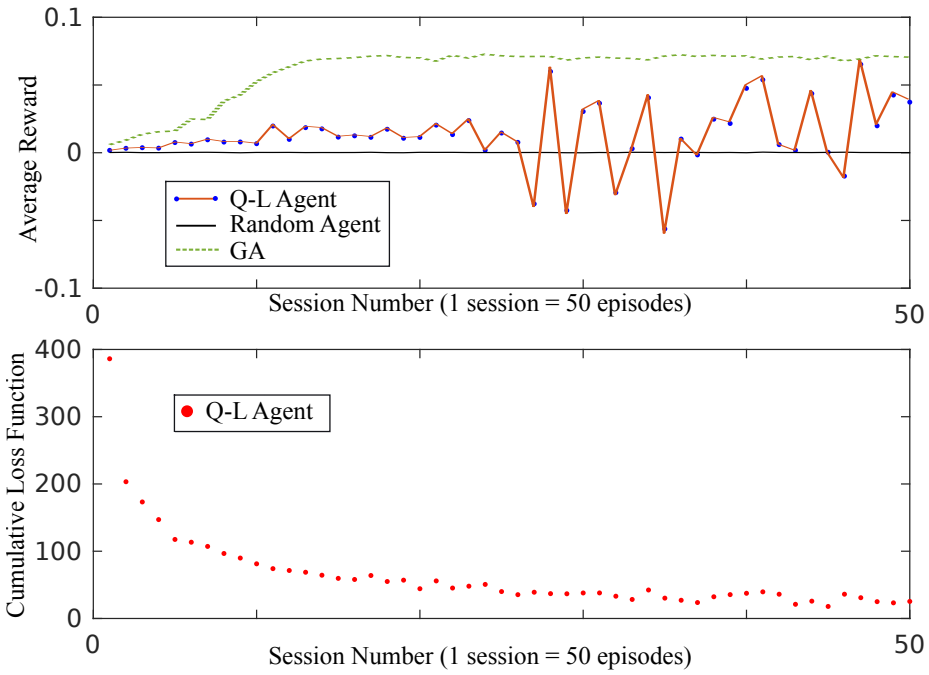

Figure 3.4: Results of initial prototype experimentation illustrating hyperparameter sensitivity (learning-rate-induced instability)

tion 3.6.1 , the RL receives the reward only when its value surpasses that of any reward received up to that point in the episode (generation session); experiments Section 3.6.2 and Section 3.6.3 receive each reward regardless of history.

3.6 Experimental Results

Brief Note on Hyperparameters Initial prototyping was performed in Matlab using a Double-Deep-Q architecture built from scratch. The effort yielded mixed success (Figure 3.4) mostly due to the abundance of hyperparameters in the algorithm and implementation-dependent subtleties. Because of our inability to achieve consistently good results, we turned to OpenAI Baselines, and Stable-Baselines for more robust implementations [98].

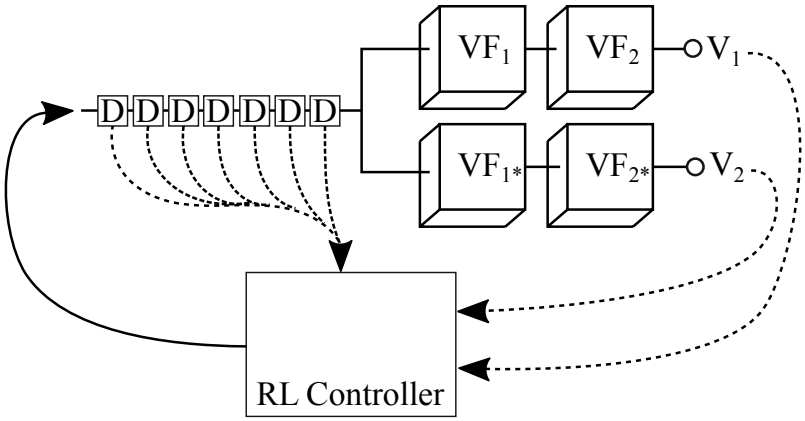

Figure 3.5: Block-level schematic of the experimental setup in Section 3.6.1

3.6.1 Volterra Models in Python

Two pairs of Volterra filters were created in a Python environment with identical coefficients drawn from a standard normal distribution. In one set of filters, two coefficients were chosen for a large binormally-distributed perturbation (+/- $0.2/\sigma$ and $\mu = 0.2/\pi$) and 10 coefficients were chosen for small perturbations (+/- $0.002/\sigma$ and $\mu = 0.002/\pi$). The coefficients of each filter were then normalized such that no coefficient had a magnitude larger than 1. We obscured the internal states of the filters from the learning algorithm, and kept the 7 most recent historical inputs as proxy states.

The stimuli were limited to 20 samples in length, corresponding to 20 steps per episode. The learning rate, α was 0.5e-4, the discounting of future value, γ, was 0.99, and the policy was ϵ-greedy with ϵ decaying exponentially from 0.1 to 0.02 over the course of 500k steps. The system was explored in batches of 32 episodes constituting a session. After each session, the network (sizes varying from 8 neurons to two layers of 64 neurons each) was retrained to approximate the updated Q values from the most recent 50k sessions.

Figure 3.6 shows a comparative study of q-learning implementation against a differential evolution implementation provided by the Scipy library.

It can be seen that the DQN stimulus generation algorithm can repeatedly synthesize stimuli at a fraction of the cost of DE once it's trained.

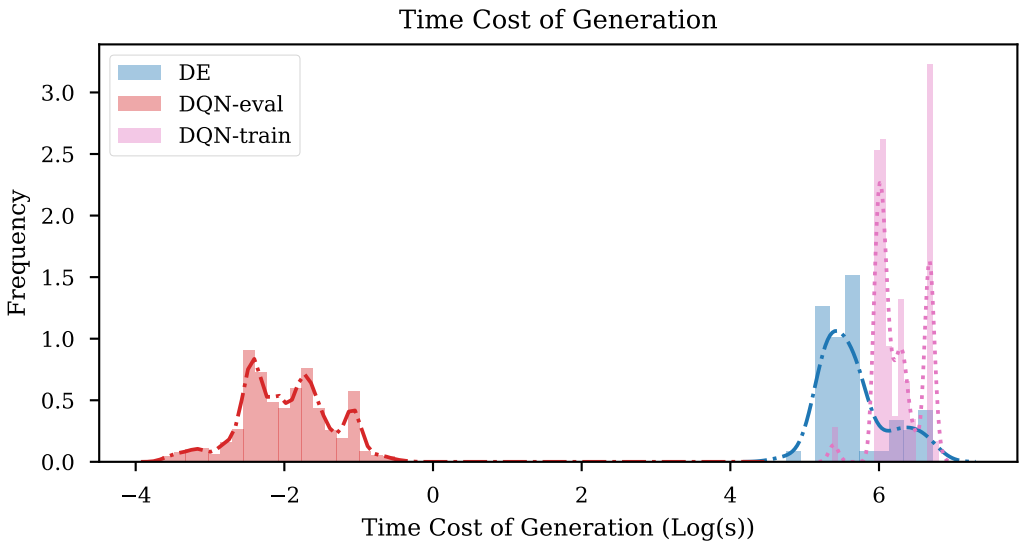

Figure 3.6: Comparison of Time Cost of Test Generation: DQN vs. DE

Table 3.1: DQN v. DE: Selected Statistics

Mean DE generation time	316.4s
Mean DE reward	8458.2
Mean DQN training time	404.1s
Mean DQN evaluation time	0.17s
Mean DQN reward	3347.4

3.6.2 Programmable Gain Amplifier in Spectre/OpenAI

Our second experimental platform was built in Pythton 3.5 and Tensorflow 1.5 and utilized the OpenAI Baselines library [97]. The implementation of DQN was based on [100].

The systems under validation were Cadence Spectre simulations. Each circuit was implemented in its own netlist, and instances of the Spectre binary were run in parallel and interacted with dynamically through our own "circuitgym," "pyspectre," and "libpsf" python libraries [101–103]. In this fashion, the circuits' inputs are manipulated and short time step transient simulations are performed.

The system under test is a 2-bit DAC feeding directly into a programmable gain amplifier with a 2-bit gain control. One version utilizes a more detailed and less idealized AHDL model for the amplifier core. Inputs are updated at 333ns intervals, and transient simula-

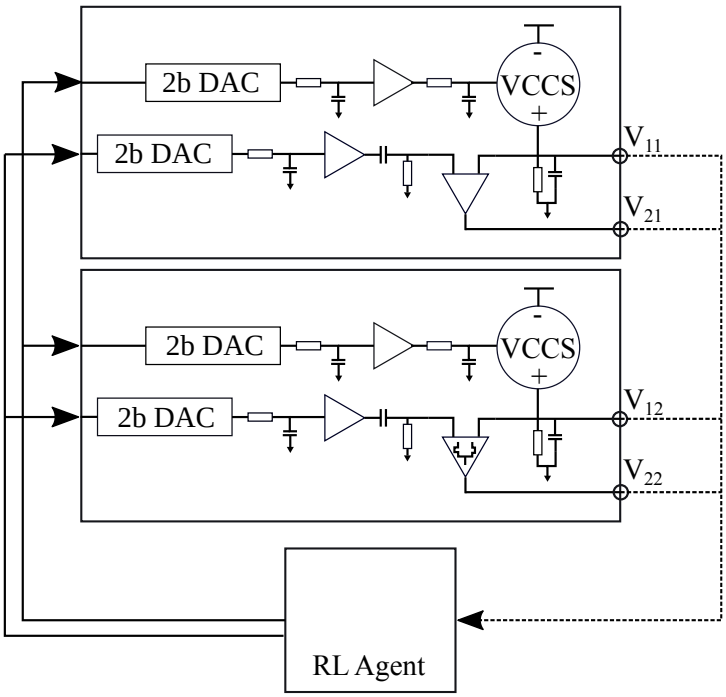

Figure 3.7: Block-level schematic of the system under validation in Section 3.6.2

tions of 333ns duration occur between each RL time step. After each time step, Spectre returns transient waveforms for all the transitional dynamics that are observable.

We exposed all 102 internal states which the two models had in common. This meant that the network must have an input dimension of (102,1). The deep Q-learning model implemented a discrete action space, and so each full input vector was assigned a probability of inducing a particular action. The circuit has 4 discrete input bits, corresponding to two 2-bit DACs. The interface between the Q-learner and the circuit was made by considering each full 4-bit vector to be a discrete action. Thus the output layer of the circuit required 16 neurons, corresponding to the 16 possible inputs. The network was configured to have two hidden layers of 64 neurons each; This configuration lead to a total of 11,793 trainable parameters in the neural model.

Three test benches were built for this circuit: a random actor, an off-the-shelf differential-evolution algorithm (DE) and our DQN algorithm. In each, a maximum of number of allowed steps was set to 12,500, corresponding to 2500 episodes, each of length 5. The

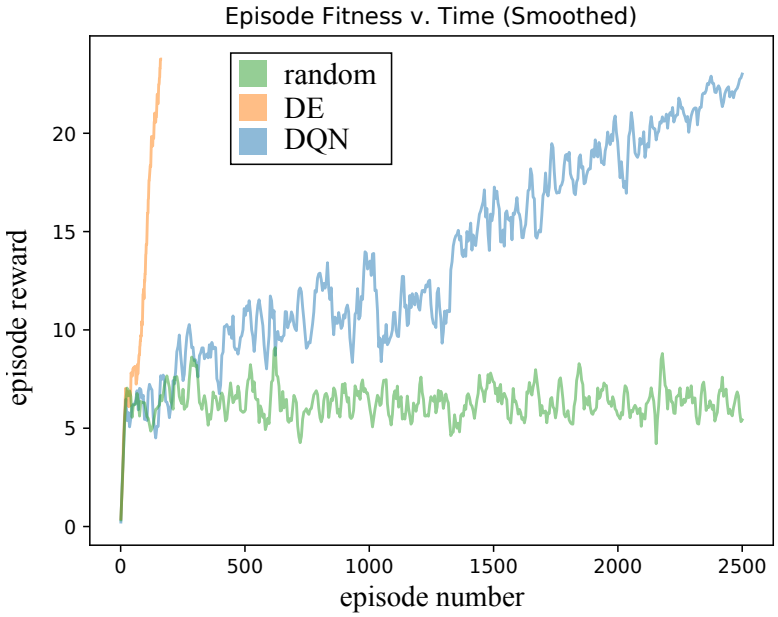

Figure 3.8: Evolution of rewards from different actors in PGA environment (Section 3.6.2)

results of the random actor exercise over time are shown in Figure 3.8 .

A more in-depth look at how the off-the-shelf DE algorithm progresses reveals a pitfall; independent trials of the DE algorithm, though the DE algorithms may sometimes converge and terminated very quickly, result in high variance in performance. A majority of the solutions proposed by the DE algorithm are poorer in performance than that from the DQN algorithm. This is a consequence of the possibility that the algorithm converges in a local solution. The DQN algorithm does not suffer from this risk because it has been analytically demonstrated to be globally optimal in the asymptotic case [92].

3.6.3 LNA Model Augmentation

In order to validate the model augmentation portion of the framework, we implemented a buggy low-noise amplifier in Cadence spectre by injecting a 10k-ohm resistor to couple two nodes which should not be directly bridged. Augmentation was performed by iteratively creating, training, and injecting SVM regressors into the bug-free model in order to reduce the observed behavioral difference. Like experiment Section 3.6.2 , the experiment was

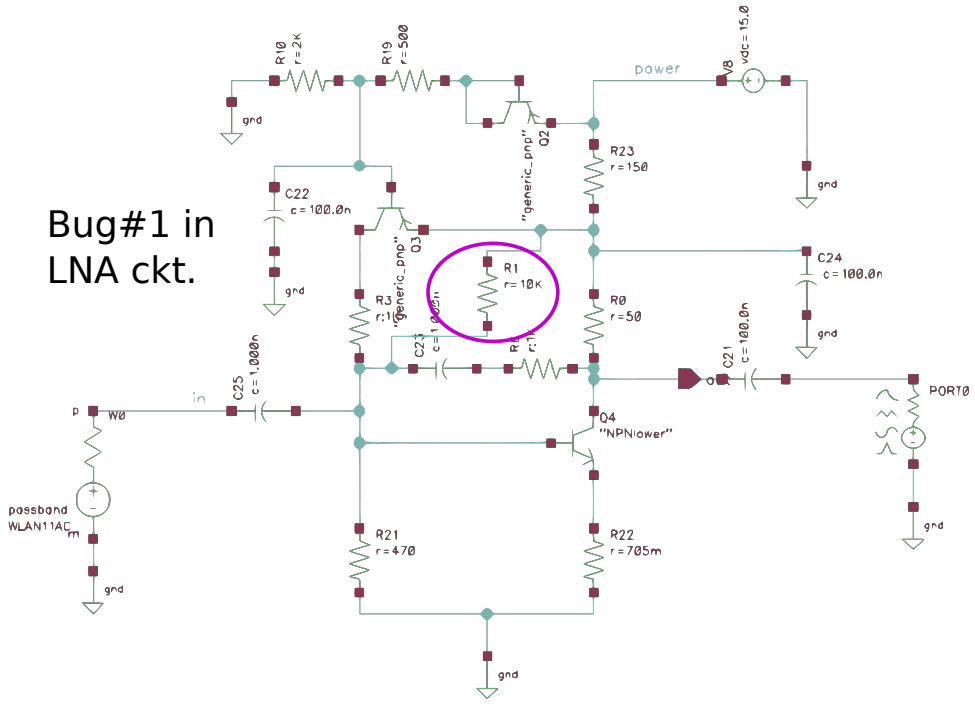

Figure 3.9: Fault-injected LNA

built in Pythton 3.5 and Tensorflow 1.5 and utilized the OpenAI Baselines library [97].

The LNA circuit is shown in Figure 3.9 ; lna model discrepancies (buggy v. bug-free), before and after 20 rounds of augmentation are shown in Figure 3.10 with light color indicating observations of high-discrepancy. Figure 3.11 plots the evolution of the mean and kurtosis of model discrepancy over all observations over the course of iterated augmentation, indicating that not only is overall error reduced, but the number of spurious, relatively high-error events is also reduced.

3.7 Conclusion

We have presented the details of our state-of-the-art reinforcement learning algorithm which serves as the stimulus-generation subsystem in a design validation framework. We have designed and conducted several experiments which leverage Q-learning to generate a stim-

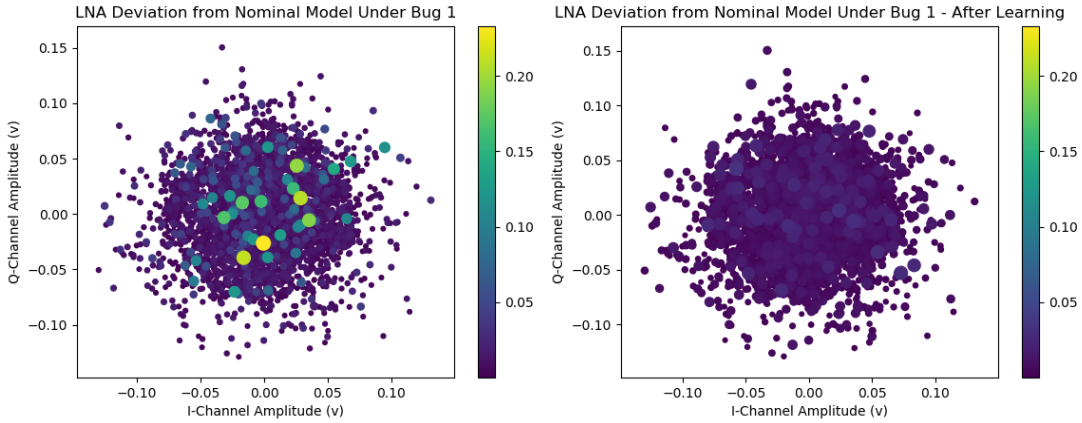

Figure 3.10: Buggy model error before and after augmentation. Lighter colors represent higher-discrepancy observations

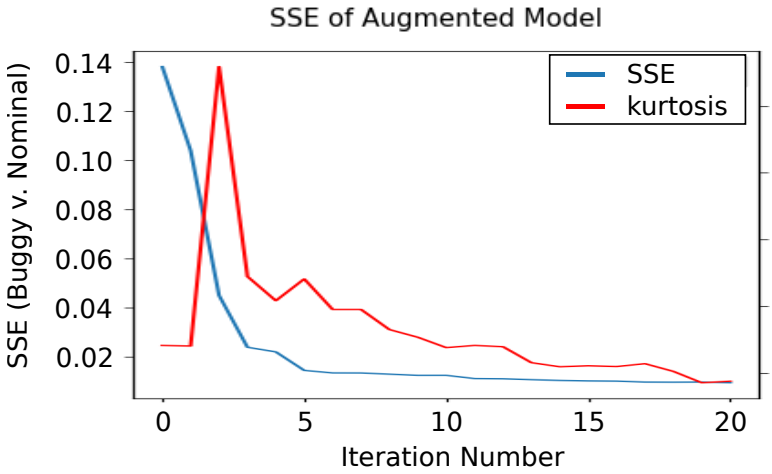

Figure 3.11: Evolution of LNA augmentation while capturing buggy behavior.

ulus which is capable of revealing a maximal amount of information about the differences between two systems. Our work suggests that reinforcement learning provides a very powerful complement to tools like differential evolution and Monte Carlo stimulus generation.

While DQN is able to ultimately outperform the other methods, it comes at the cost of computational time (about 10x more). It however outperforms the optimization solver from a warm-start by a factor of roughly 1800x, suggesting that if appreciable restarts are foreseen, the benefits of RL may outweigh its up-front costs.

Some validation tasks can be accomplished with traditional specification tests (i.e. two-tone test) while others might require slightly more entropic stimuli like white noise in order to excite behaviors of interest. Circuit complexity, situation, or mission-criticality may mandate running the additional number of simulations required to train a high-performance agent to reveal the global solution, something that stochastic nonconvex optimizers cannot provide. We will continue this work, looking at broader classes of circuit and varying degrees of bug magnitude and nature. Additionally, we will explore the limits of RL's warm-start advantage.

CHAPTER 4

THE DESIGN-OF-EXPERIMENTS APPROACH

4.1 Introduction

In hierarchical system design (SoCs, SoPs), models for mixed-signal components (e.g. regulators, data converters, I/O, RF) are described at the behavioral level (Simulink or MATLAB) with the complete system described as an interconnection of these models. Typically, simulation of a complete design at the flat netlist level is prohibitively compute-intensive. Hence, component level behavioral models are used for system simulation, dropping down to netlist level simulation only for those modules for which accurate behavioral models are not available. As a consequence,the accuracy of the behavioral models used, across the entire space of their operation (time or frequency domain), is of primary concern. At every step of the design, the behavioral models must sufficiently match the input-output behaviors exhibited by the corresponding transistor level netlists or implemented silicon, and any disagreement, or error must be quantified. Note that these errors can stem from behavioral modeling inadequacies or design bugs at lower levels of design (transistor level or implemented silicon), either of which will cause disagreements between behaviors exhibited by higher and lower level descriptions of the system and its constituent modules. Regardless of the source of the error, the relevant behavioral level models need to be re-built or amended to accurately reflect low-level circuit design behavior. This is necessary to determine if all design specifications are met at the system level and that no spurious system level behaviors are possible. Such model-building is also necessary for design diagnosis purposes [37], to identify the source of the error and generate design fixes.

4.2 Background

Contemporary system designers find themselves spending increasing quantities of resources on design debugging, functional verification, and post-silicon validation [1] and [93]. While the analog specifications of modules at two different levels of the design can be checked for equivalence, such a check does not guarantee behavioral equivalence across the entire space of input stimuli because the set of specifications used to check for equivalence may themselves be *incomplete* [52, 84]. For similar reasons, formal methods for design equivalence checking may fail because designer inserted assertions may not cover all input-output behaviors and are generally myopic and constrained to specific input conditions.

The use of rapidly exploring random trees (RRT) for analog circuit test generation was presented in [85, 86]. The idea was to quickly explore all reachable points in the state space of the analog circuit. Our approach, in contrast, progressively focuses on those points in the state space of the analog circuit where differences exist between the behavioral description of a circuit and its netlist (or silicon) level implementation. The work of [89–91] was among the first to combine formal verification methods with simulation driven techniques to explore the limits to which analog design specifications can be stressed under multi-parameter circuit level perturbations without causing specification violations. Our objective in the present research is not limited to device specifications but is applicable across the entire space of possible input stimuli to the system.

There has also been past work on test generation driven analog design validation [37, 58, 104]. The approach of [37, 58] focused on finding stimulus that maximized the difference between two representations of the same system (e.g. behavioral vs. netlist), learning and correcting for the difference using model-augmented learning kernels and repeating the process until no further differences are excited via stimulus generation. This approach incurs *large numbers of learning iterations* for hierarchical system descriptions in which differences exist between the two design representations above, across wide domains of

the behavioral space (*even due to a single design bug*). The approach of [104] uses reinforcement learning algorithms to excite and learn about response tail behaviors to the time and frequency domain statistics of input stimulus. This converges to globally optimum solutions after multiple learning iterations as opposed to the genetic stimulus generation approach of [37], but is very slow, running into days of computation on a modern personal computer for simple circuits such as amplifiers.

4.3 Relevance to Prior Research and Key Contributions

Prior design validation algorithms (genetic evolution based [37], reinforcement learning based [104]) used complex compute-intensive algorithms to design time-domain stimulus to emphasize temporal and statistical differences between the response of the designs at two different levels of design abstraction to the stimulus. These differences were used to train learning kernels designed to augment the high level model and compensate for the differences observed. The process of test stimulus generation and behavior compensation was repeated iteratively (called learning iterations) until no further differences above, could be excited.

Key Contributions: As opposed to prior algorithms, instead of optimizing a single test stimulus for each iteration of kernel based learning, a *population of stimuli is evolved between each learning iteration* (note this is different from genetic evolution [37] in which a single optimal stimulus is produced from a population of stimuli in each learning iteration). This significantly reduces the burden of test stimulus optimization in [37, 104], while allowing the learning kernels to learn across the statistical characteristics of temporal and frequency domain device responses to an *entire population of stimuli* rather than a single stimulus in each learning iteration. Note that the majority of circuit simulation runs in the test optimization steps of [37, 104] are wasted (i.e. they do not contribute to behavior compensation by the learning kernels in each learning iteration). In the current approach, every circuit simulation run counts towards the overall behavior difference learning process.

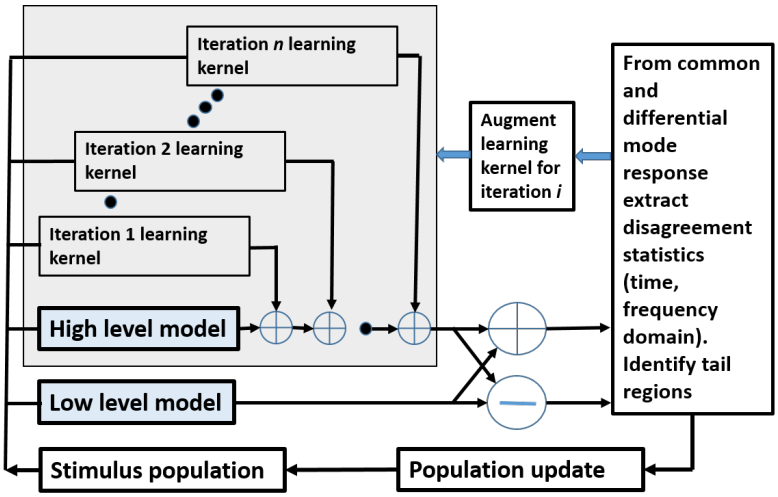

Figure 4.1: Overview of proposed validation methodology.

Repeated observations of the dynamics of a pair of systems stimulated in unison (high vs. low level circuit description) are made across the population of stimulus considered. A statistical model describing their disagreement is constructed. The tails of the disagreement distribution represent high-risk, low-occurrence corners of the system's operational space. Our approach rapidly identifies the tail-regions of this distribution and generates test stimuli whose simulations are likely to reveal valuable new disagreement data, via simulation, coming from the highest-risk regions of the tail. The population of test stimulus is evolved to include the generated stimuli and used to train the learning kernels as before. As further learning iterations occur, the tails of successive disagreement distributions are eliminated until they become normal distributions with virtually zero variance. *This results in significant validation time speedup while maintaining high coverage of modeling inaccuracies and design bugs*.

4.4 Approach

Figure 4.1 gives an overview of the proposed validation approach. A database of observations consisting of the common- and differential-mode responses of the two models is built through a bootstrapping technique. In order to begin constructing an observation

database, we must first estimate m, the amount of memory present in the system so that our state-space model will be of sufficient dimensionality. Based on the result of a memory-estimation procedure (Step 1 in Figure 4.1 and detailed in Section 4.4.1), the observations in the database can be decomposed into $< state, input, response >$ triplets. Step two is bootstrapping, wherein batches of uniformly random stimuli (transient time domain waveforms consisting of piecewise-linear segments) are used to concurrently stimulate both the high and low level models of the analog circuits being validated. Initially there are no learning kernels present in the high level model. Statistical density models of the observed common-mode and differential-mode states and responses in the database are constructed. After each batch of random stimuli is simulated and the observations are collected, decomposed, and added to the database, we update the density models.

Because any learning is entirely dependent upon training data, and we are primarily interested in the ways our models differ, we must insure our database contains a sufficient proportion of this useful information. If the models are highly similar, then it follows that the majority of observations are of little use. At step 3, we quantify the proportion of high disagreement observations in our data. To gather additional *useful* observations, in step 4, we sample from the tails of the differential-mode density model (the regions of the state-space where we observe the greatest disagreement between the models), project stimuli likely to drive the system into those regions, and simulate them. As the additional simulations are conducted, the statistics of disagreement across all observations in the database are recalculated. Finally, in step 5, we enhance our high-level model by training a machine-learning network on our data and attaching it in parallel with the model. The database is then updated to reflect the respose of the updated model, and its statistical models are rebuilt. The process is repeated, augmenting the high level model with learning kernels in each learning iteration until the disagreement statistics approximates a normal distribution with a very low (specified) variance. At this point the algorithm terminates with the automatically generated augmented high level model matching the behavior of the low level

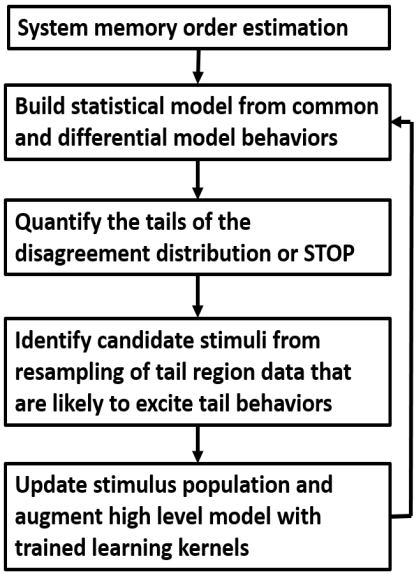

Figure 4.2: Validation methodology: algorithmic steps

model across its entire input space.

Preliminaries: The entire suite of algorithms developed is structured around making repeated joint observations on two (or more) systems, each modeled as a partially-observable Markov decision process (POMDP) [105], with the goal being to generate the largest proportion of useful information in as efficient a manner as possible. The POMDP formulation requires the careful discretization of time: a period is chosen for which input signals are changed and the states of the system are recalculated (a continuous-time transient simulation can occur within a timestep). Each input signal must begin at a known reset condition and must span a sufficient number of time periods for the constituent systems to enter new states. At which time an additional input is given, and the responses are recorded. The entire observation, $\boldsymbol{O}_i$, is thus recorded as a 3-tuple composed of: 1) the system state estimate, $\boldsymbol{S}_i$, a vector of length $mb_i + (m-1)b_o$ (b_i representing dimensionality of inputs; b_o, dimensionality of outputs); 2) the most recent input, $\boldsymbol{A}_i$, ("action" in MDP parlance); and 3) the system's response, $\boldsymbol{R}_i$ (output in the most recent time step). Additionally, we define an input vector, $\boldsymbol{x}$, composed of $\boldsymbol{S}$ and $\boldsymbol{A}$, and a response vector, $\boldsymbol{y}$, equivalent to $\boldsymbol{R}$: $\boldsymbol{O}_i \coloneqq \{\boldsymbol{S}_i, \boldsymbol{A}_i, \boldsymbol{R}_i\}$, $\boldsymbol{x}_i \coloneqq \{\boldsymbol{S}_i, \boldsymbol{A}_i\}$ and $\boldsymbol{y}_i \coloneqq \{\boldsymbol{R}_i\}$.

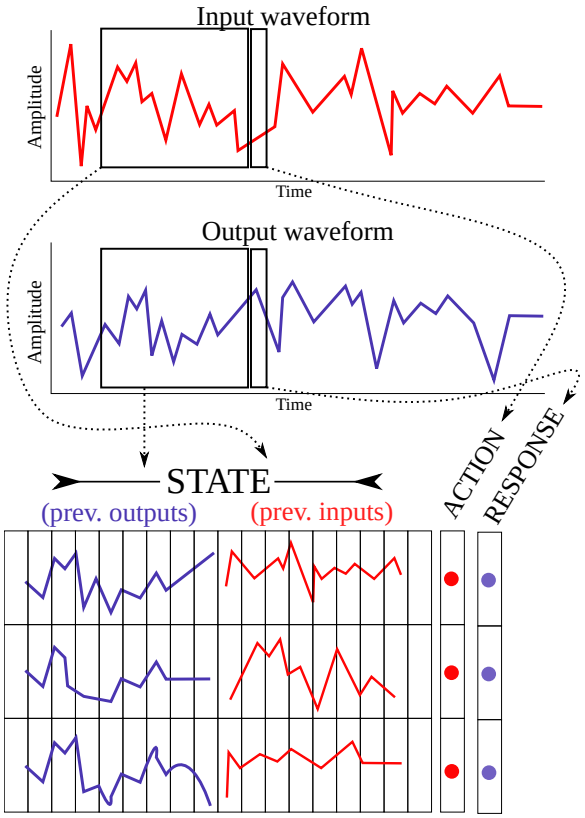

Figure 4.3: Composition of Observation Database

In this fashion, arbitrary input and output observations (length n) of a system with hypothetical memory m can be decomposed into n experiments by convolving a window of length m across the input and output time-series, and recording the y value corresponding to time point $[m+1]$ relative to each window. Figure 4.3 illustrates the concept.

4.4.1 Memory Estimation Algorithm

In this algorithm, we attempt to measure the length of a pseudo impulse response for the system described by $\mathcal{S}_2(x) - \mathcal{S}_1(x)$. We assume a non-autoregressive, or finite-impulse-response behavior. Because the system may be highly nonlinear, an observed impulse response will be a function of both the state of the system at the time of application, and the magnitude (and slope) of the impulse. In a high-dimensional setting, this creates the problem of having to explore an infinite number of potential impulse responses all over the state space with varying impulses. First, we create many paired stimuli for experimentation; the individual stimuli of the pair are identical with the exception of the input sample at time k+1 where k is the maximal possible memory exhibited by either system. Both systems are first excited with stimulus A and then both are excited with stimulus B. Many such paired experiments are conducted, and the difference signals in response to stimulus B are observed at all nodes and are subtracted from those observed in response to stimulus A; for each observation made in the range $(k+1, 2k)$ the hypothesis tested (using the 2-sample Kolmogorov-Smirnov test) whether the data in that time-step comes from the same distribution as the data observed before time $k+1$, when the systems were exposed to the perturbed input sample.

More batches are simulated in a similar fashion, each resulting in a length-i vector of p-values for the hypotheses that observations at time-step i are indistinguishable from unperturbed data. After several batches are complete, we create a kernel density estimation for the distributions of p-values within each hypothesis. As additional batches are calculated, the kernel density model is updated, and the Kullback-Leibler divergence is calculated be-

tween the old density model and the updated model. The algorithm terminates when the distributions of p-value for each hypothesis are well understood:

$$\frac{\text{KLD}_0}{\int\limits_i \text{KLD}_i} < 2\% \tag{4.1}$$

4.4.2 Bootstrapping and Density Model Building

With a good estimate of the memory order of the difference system, we begin repeating individual experiments of length $m + 1$ using a uniform random stimulus as input. During these experiments, we build a kernel density model of the output distributions that we observe from the paired systems, both common-mode: $\frac{1}{2}(\mathcal{S}_2(x) + \mathcal{S}_1(x))$, and differential mode: $\frac{1}{2}(\mathcal{S}_2(x) - \mathcal{S}_1(x))$.

Increased observability of internal nodes in the models adds dimensionality to this model, which includes all covariance terms. We continue to make observations and update our model in batches, while measuring the Kullbeck-Leibler divergence between successive updates of the density models between batches. The algorithm terminates with the K-L divergence between density models falls below a percentage of the accumulated sum of K-L divergence of all updates in a fashion similar to Equation 4.1 . In this way we ensure we have a minimally thorough sampling of both the state-space and the kinds of disagreement the models can produce.

4.4.3 Tail Identification

After having done a cursory uniform sampling of the differential system statistics, we have to identify 1) whether there is "excess kurtosis" in the system and 2) the boundary of those regions which exhibit high-consequence and low-likelihood error. In order to distinguish between the "head" regions and "tail" regions, we perform a piecewise linear fit to the sorted error of all bootstrapped observations. As many segments as are necessary to keep the sum-squared error under 5% are used. The segment corresponding to the greatest

amount of disagreement is used to represent the range of observations coming from the tail.

4.4.4 Resampling from the Tail in Transformed Space

The existence of excess kurtosis and "tail" regions by definition implies that our bootstrapping approach with uniform sampling will yield relatively little information from these regions. It is also true that these regions are also those of the most interest to us in the task of validation. This section explains a strategy for efficiently and intelligently designing subsequent experiments so as to maximize the expected value of each and every simulation from this point forward. The concept of Kriging is used, but is significantly adapted to accommodate the potentially very high-dimensional nature of our data in validation.

First, we establish the desired number n of additional experiments to conduct per batch. Next, we define a neighborhood size, b. In this work, we used $3^(m + 1)$. Next, we sort the observations (in the tail) by decreasing discrepancy, and operate over the first n data points. For each data point, we identify b data points which are adjacent in the state-space. Next, we perform a PCA decomposition of that region of the state-space, centered at data point i, and keep only the first 2 components. We then perform traditional Kriging inside the reduced-dimensionality PCA transformed neighborhood of the state space surrounding our point of interest.

Repeating this process n times yields n points in the region of the tail which should be simulated in order to yield additional information about the dynamics of the discrepancy in that region.

4.4.5 Model Augmentation from Tail Data

At this point, we have a sufficient volume of information coming from the regions which are both the most interesting and the most challenging to excite. Using this information, we train a machine-learning regression model to translate the output of System 2 into that of System 1, thereby completing the goal of identifying and learning from model deficiencies.

In this work we have used an SVM regressor to perform a correction of the output of System 2.

Once the regressor is trained, bootstrapping begins again, as the resulting model is now fundamentally different, and the statistics have to be re-explored from scratch. There are two alternative termination conditions: 1) The resulting corrected model is now approximately normal and homoscedastic and no more exploration is required; at this point diagnosis can begin to move corrective structures and parameters into the interior of the model. Alternatively, 2) The chosen ML regressor is unable to learn high-complexity relationships; continued learning will be limited and we must revisit assumptions of memory-length and of ML complexity.

4.5 Experimental Results

4.5.1 Experimental Results of Memory Estimation Algorithm

One hundred trials were conducted, whereby a Volterra filter of memory length 5 and nonlinearity of the third order was constructed at random. One coefficient of 84 was selected at random for perturbation. The coefficient was perturbed by multiplying it with a draw from a bigaussian distribution. Thus the paired system is created. The ground-truth memory is taken to be the oldest of the time-delayed inputs present in the nonlinear product associated with the perturbed coefficient.

4.5.2 Experimetnal Results in a Low-Dropout Power Regulator

A Low-Dropout linear power regulator (LDO) was constructed around an opamp-based servo circuit. The LDO was instantiated three times. And simulated with 1000 stimuli constructed from a uniform distributions. In one case, an ideal opamp represented with a Verilog-A model was placed in the circuit. In the second case, a more sophisticated, less ideal LDO was used, and in the third case, a transistor-level op-amp circuit was used. The top-level netlist is depicted in Figure 4.6 .

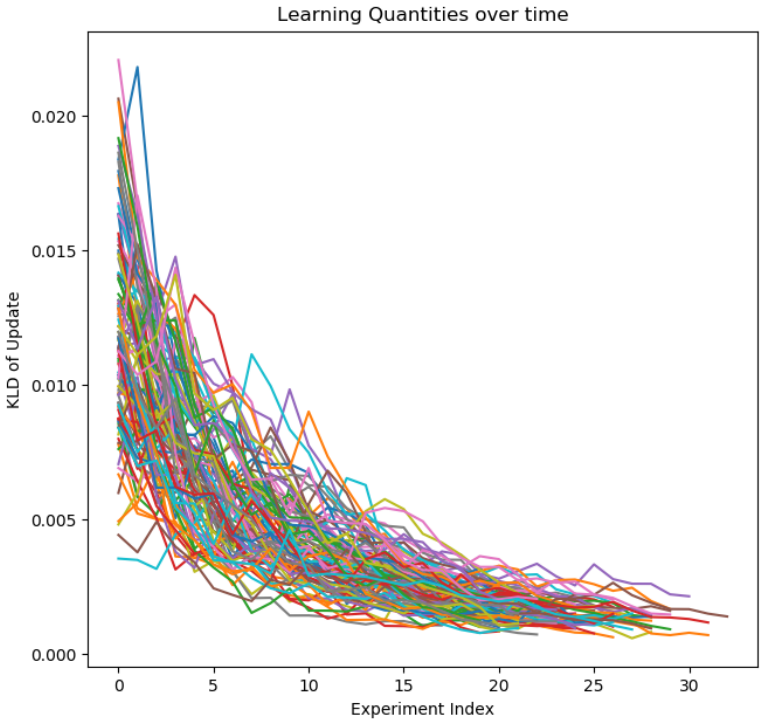

Figure 4.4: Kullbeck-Leibler Distance between successive KS-test P-values across 100 trials.

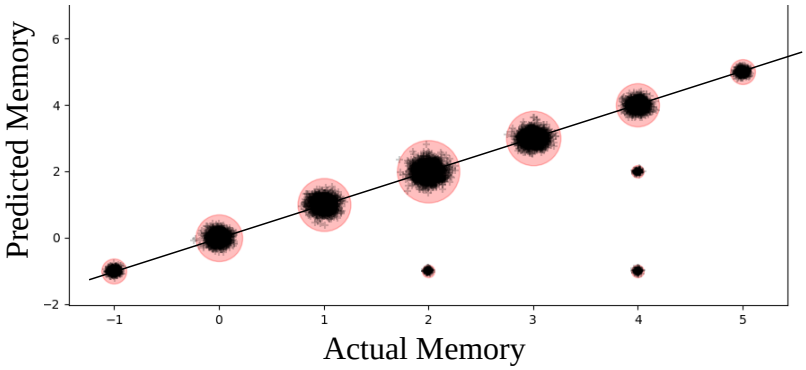

Figure 4.5: Prediction accuracy of memory estimation algorithm in prediction the memory depth of the largest coefficient-difference in 1000 random Volterra filter pairs.

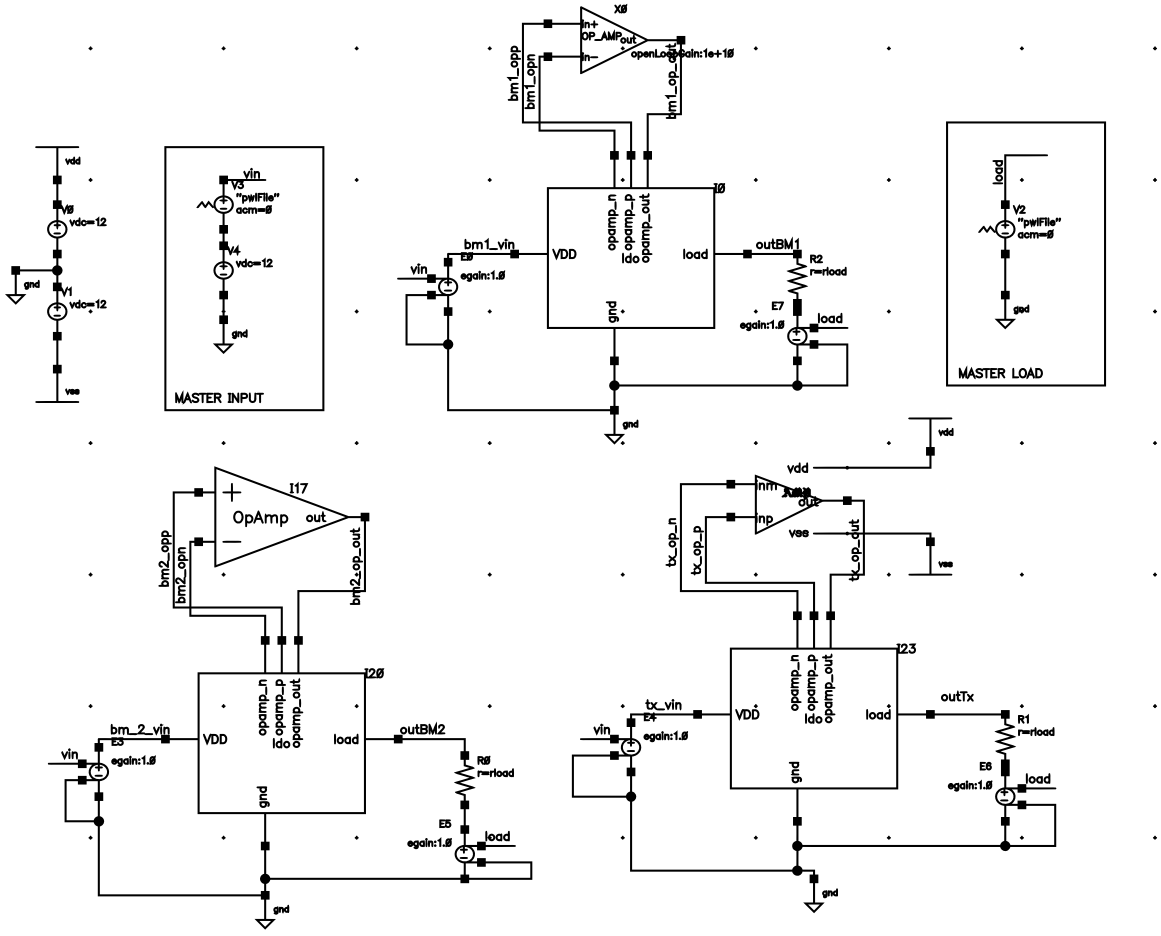

Figure 4.6: Top-level netlist of 3 LDOs, with different op-amp models: ideal Verilog-A, nonideal Verilog-A, and transistor level.

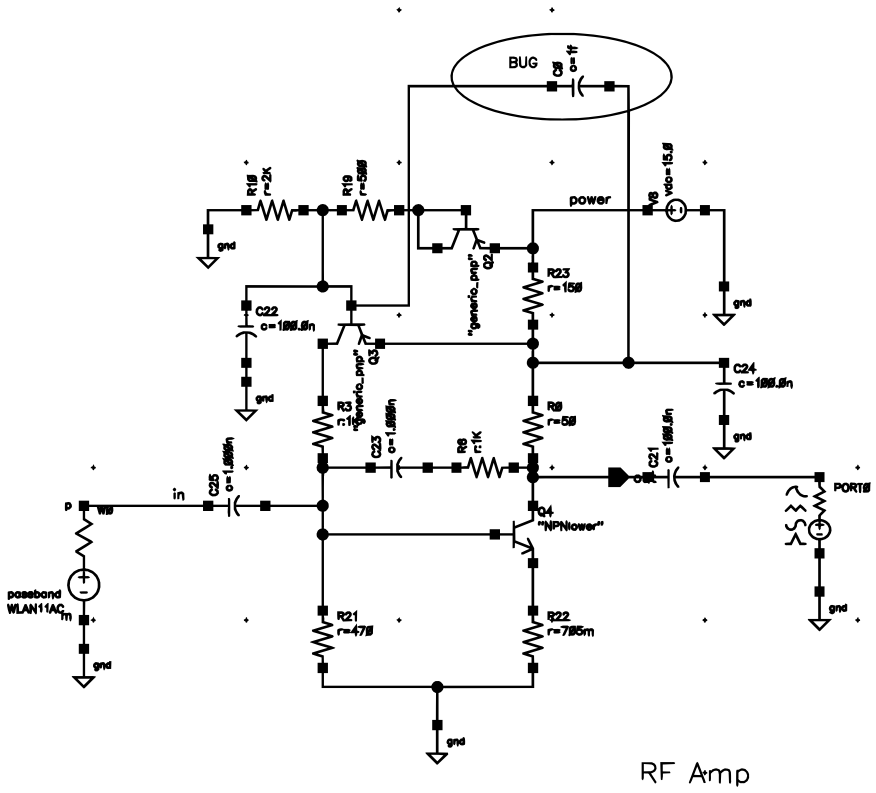

Figure 4.7: Buggy LNA Circuit

The more advanced Verilog-A model has a much more compact distribution, though both models exhibit high degrees of kurtosis.

4.5.3 Experimetnal Results: Augmenting a Transistor-Level LNA

A transistor-level LNA model was simulated alongside three instances of itself, each of which manifested various design bugs (injected resistors, capacitive coupling, etc). The circuits were all simulated in a shooting-mode envelope simulation with random 802.11 data. Iterative behavioral model augmentation was performed, and the buggy behaviors were all learned and brought in-line with the transistor model.

Figure 4.7 shows the LNA circuit with an injected "bug." Figure 4.8 shows a portion of the quadrature channel envelopes in the time-domain in response to 802.11 excitation before learning, and Figure 4.9 , Figure 4.10 , and Figure 4.11 all show the same portion of signal after the buggy models have been learned and the nominal models augmented.

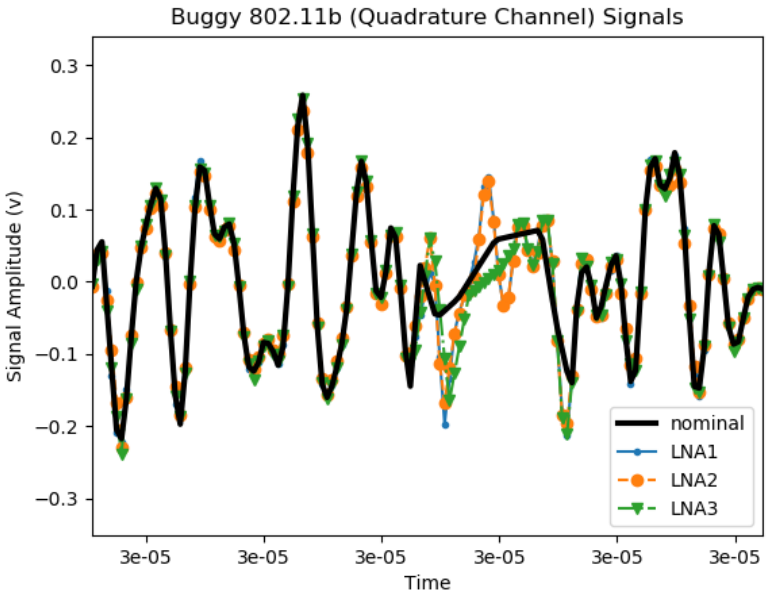

Figure 4.8: Quadrature signal envelopes of nominal and 3 implemented LNA circuits in response to IEEE-802.11 data - Before Model Augmentation

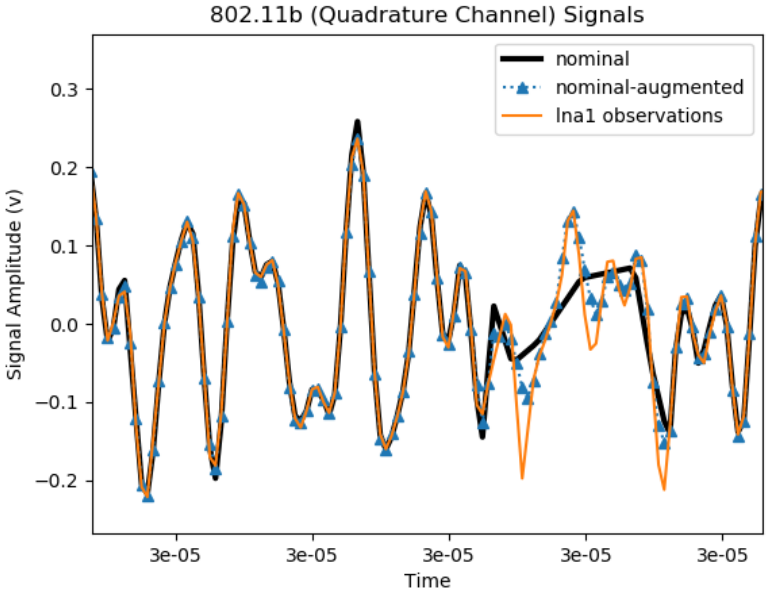

Figure 4.9: Quadrature signal envelopes of nominal, augmented-nominal, and implemented LNA-1 circuits in response to IEEE-802.11 data

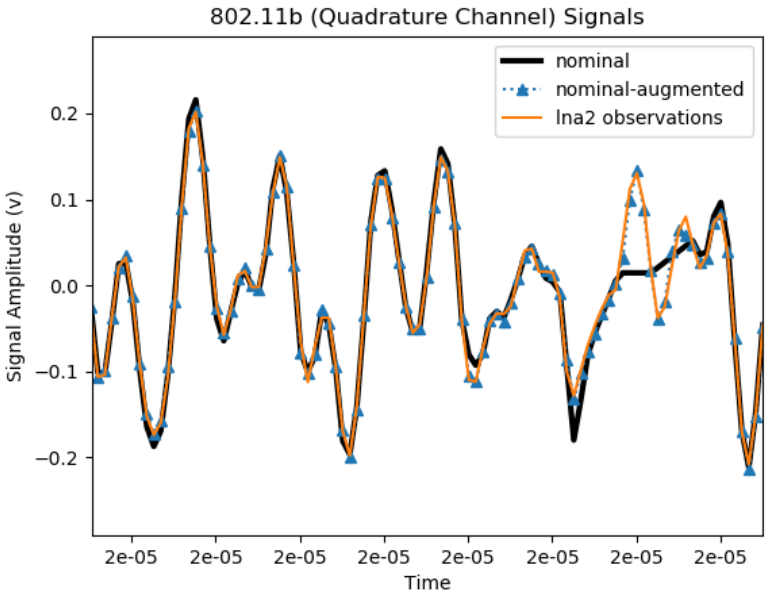

Figure 4.10: Quadrature signal envelopes of nominal, augmented-nominal, and implemented LNA-2 circuits in response to IEEE-802.11 data

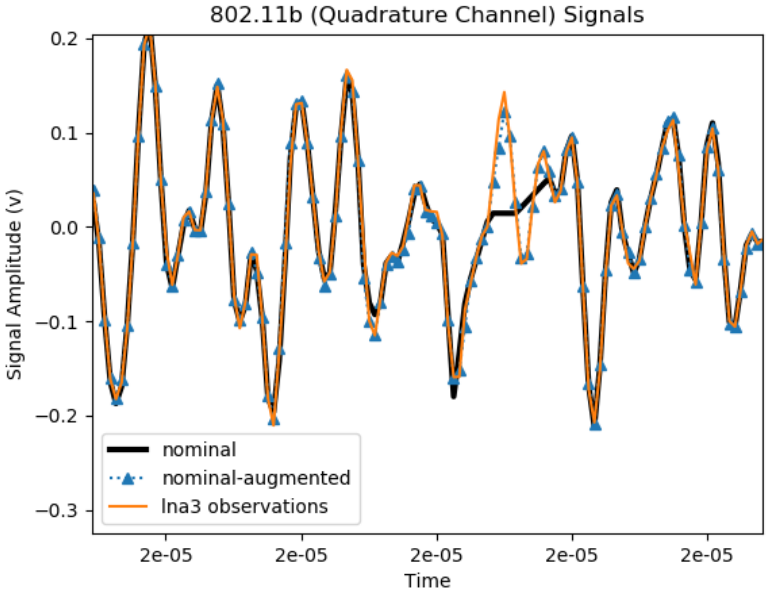

Figure 4.11: Quadrature signal envelopes of nominal, augmented-nominal, and implemented LNA-3 circuits in response to IEEE-802.11 data

4.5.4 Experimetnal Results in a Hardware RF Transceiver Front-End

A radio-frequency transceiver was assembled using commodity evaluation modules including an arbitary waveform generator, a direct up-conversion mixer, a power amplifier, a 20dB attenuator, a direct down-conversion mixer, and a high-speed digitizer. The mixers shared a phase-coherent LO at 2.4GHz. A naiive behavioral model was created using common weak-nonlinearity model parameters for linear gain, and IIP3. The hardware system was then operated under 35 different biasing conditions by varying the PA supply and the up-conversion mixer supply individually between 2.5 and 5.0 volts in 0.5 volt increments. In each case of suppressed bias conditions, performance is expected to degenerate; and in each case, the naiive behavioral model is expected to be increasingly insufficient to describe the observed dynamics as biasing conditions deteriorate and the devices begin to enter cutoff. For each of the 35 instances, the naiive behavioral model was augmented in attempt to capture the dynamics observed in the hardware system. Model error magnitudes are presented throughout the progression in Figure 4.12 (n.b. the color scale is logarithmic error). In Figure 4.13 , the time-domain response of the nominal model is contrasted to the nominally-biased hardware. In Figures Figure 4.14 and Figure 4.15 the time-domain hardware observations under worst-case bias degradation are contrasted with model output before (Figure 4.14) and after (Figure 4.15) model augmentation.

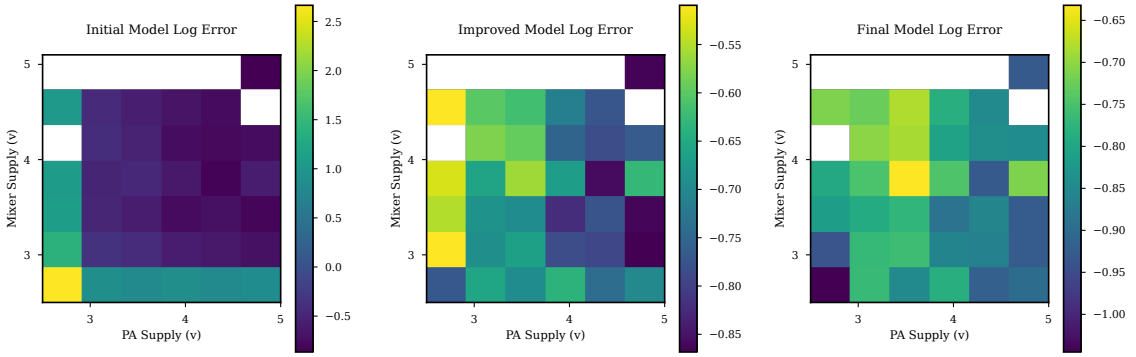

Figure 4.12: Performance of 35 bias-compromized hardware extracted models over the course of learning.

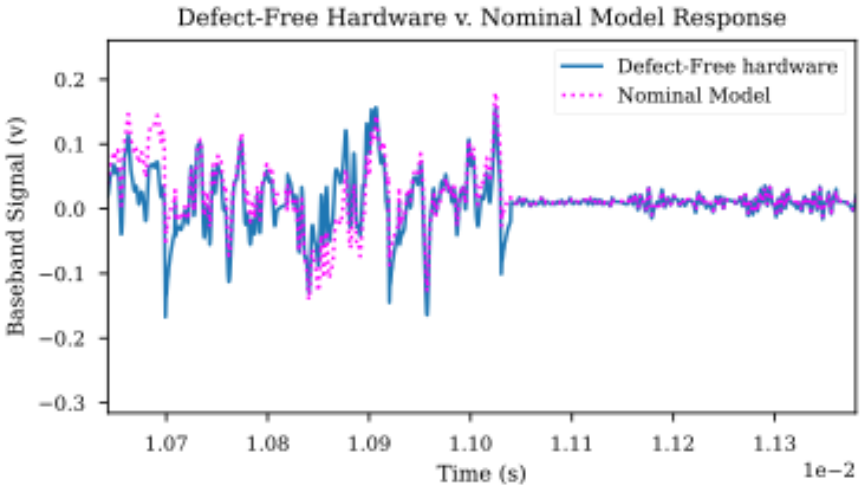

Figure 4.13: Time-domain waveforms of nominally biased hardware and nominal behavioral model.

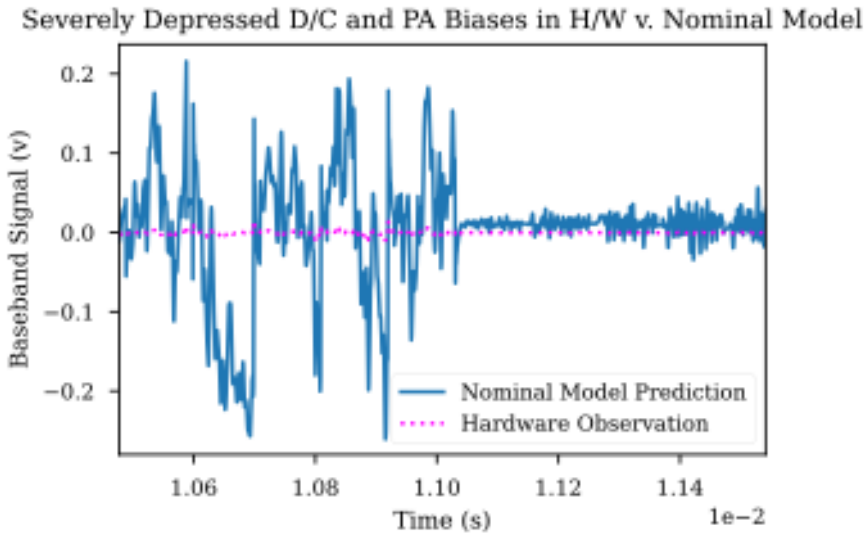

Figure 4.14: Time-domain waveforms of worst-case biased hardware and nominal behavioral model.

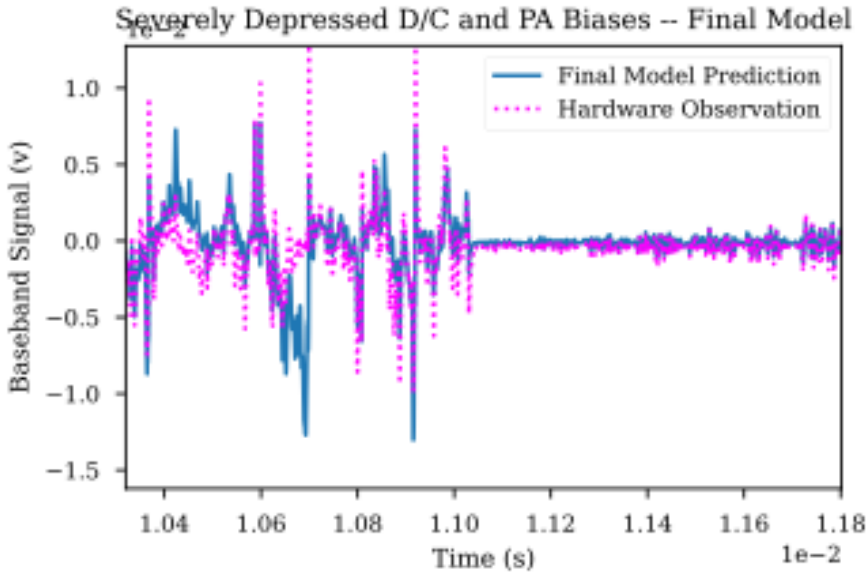

Figure 4.15: Time-domain waveforms of worst-case biased hardware and nominal behavioral model after completion of model augmentation.

CHAPTER 5
DIAGNOSIS

5.1 Debugging AMS Systems Through Iterative Learning

In this section, I present a novel technique for the algorithmic formulation of circuit diagnosis predictions which does not require any assumptions about the nature of design errors [37]. In the modern mixed-signal SoC design cycle, designers must assess equivalence of module descriptions across different levels of hierarchy, e.g. behavioral vs. transistor level descriptions or transistor level descriptions vs. fabricated silicon. If such differences (anomalies) are detected, then diagnosis is concerned with identifying the module in a hierarchical design description of the system that is most likely the root cause of the anomaly. Previously proposed machine-learning classifiers require prior knowledge and assumptions about the kinds of design errors likely to be encountered.

Our method employs iterative and alternate on-the-fly test generation and fitting of embedded low-order nonlinear filters to produce a best- guess estimate of the root cause of the anomaly. Experiments are conducted on two test vehicles in simulation, an RF transceiver and a phase-locked loop, several bug models are implemented, and the systems diagnosis predictions are analyzed.

5.1.1 Algorithms

The presumption of this work is that the designer or test engineer has a hierarchical model of the system and that either simulation results or a silicon device has been produced which behaves fundamentally differently from other behavioral descriptions *and* has failed specification tests. One subsystem within the DUT contains the root-cause of the failure, and engineers wish to know which module has failed and requires further attention.

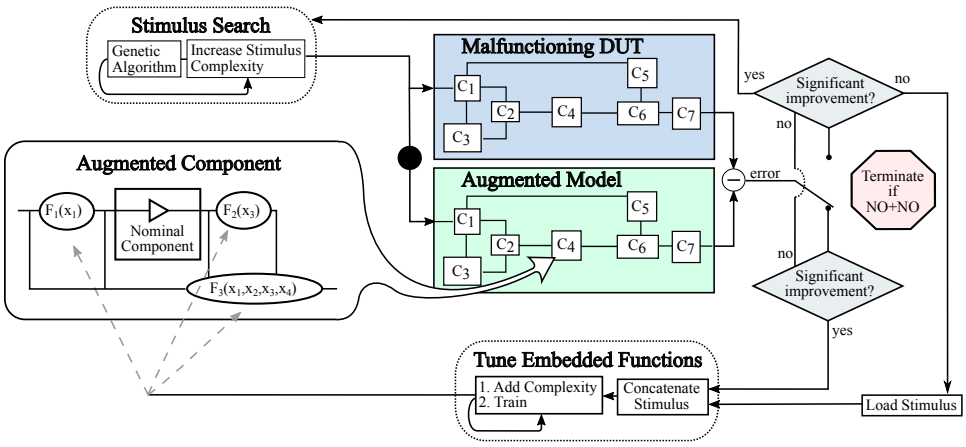

Figure 5.1: Diagnostic Algorithmic Approach

A variety of test stimuli generated using a stochastic search algorithm as shown in Figure 5.1 are applied simultaneously to both the DUT (low-level description) and its simulation model (high-level description). The outputs of both are recorded. Several hypothetical system models are created by copying the system model n times and iteratively augmenting individual constituent modules, $C_1, C_2, C_3...C_n$, with embedded neural learning structures. All the hypothetical models are trained using the previously obtained stimuli, and test stimulus generation is performed again to maximize the error between the DUT and the trained (updated) hypothetical models. Diagnosis rests on the assumption that the *residual is minimum only when the augmented component corresponding to the buggy module is trained and is not minimum when training is performed on augmented components of non-buggy modules.* So, hypothetical models are ranked in terms of the error observed between the DUT and hypothetical model response. This rank serves as an estimate of the relative likelihood that the root-cause location of a bug in the DUT corresponds to the augmented component within the hypothetical model.

5.1.2 Completed Experiments

RFTX (Supply Coupling between PA and LNA): In this experiment, software models are used for both the design-intent and the buggy system. The bug model in this experi-

ment is the inclusion of unintended supply coupling between the PA and and the low-noise amplifier (LNA), whereby large current demands of the PA result in an instantaneous decrease in the LNA's linear gain. The RF system is modeled in a baseband equivalent form, with only the in-phase channel being used for testing. Polynomial models are used for the preamp and LNA stages, and a Saleh model for the PA stage. Though the algorithm misdiagnosed the preamplifier in this experiment, the LNA was a close second. The ruling out of the mixer and the PA is still a useful precursor to other approaches, as the candidate pool was, in this case, reduced in size by half.

Integer-N PLL (Bug in VCO): In this experiment, an integer-N PLL design is debugged. Simulation models are used to represent both buggy and design-intent systems. The voltage-controlled oscillator (VCO) bug models nonlinearity in the oscillator with the introduction of an even-order distortion component present in the output of the oscillator; the magnitude of the distortion is proportional to the control voltage. Thus, larger control voltage magnitudes lead to a greater amount of distortion. This could be plausibly witnessed in silicon or in a poorly designed VCO. The results are shown in Figure 5.4. It is interesting to note that the VCO was correctly diagnosed, but the comparator was not far behind. It would make sense that the two could elicit similar behaviors, given that the bug could be viewed as existing either at the VCO's output, or at the comparator's input. Even order distortion will introduce very high frequencies (which are likely lost in the LPF), and very low frequencies, which could be explained by an offset present in the comparator.

Integer-N PLL (Bug in LPF): In this experiment, a bug is introduced into the LPF through the introduction of even order distortion at the filter's output. The results are shown in Figure 5.3. In this case, the LPF was correctly diagnosed, though by a small margin. Again, it is interesting to note that it was the XOR block whose resulting penalty value put it in second place for diagnosis. One can reason that its proximity to the LPF would encourage such behavior. Also, it may be that because there are distinct signal do-

mains, the algorithm can distinguish between the high-frequency oscillator domain and the low-frequency control voltage domain with ease.

5.1.3 Conclusion

In this work, we have presented a methodology for the automated detection and diagnosis of design errors in analog/mixed-signal/RF circuits. Most notably, without an assumption having been made as to the nature of design errors, we have demonstrated cases of successful automated diagnosis of bug root-cause location.

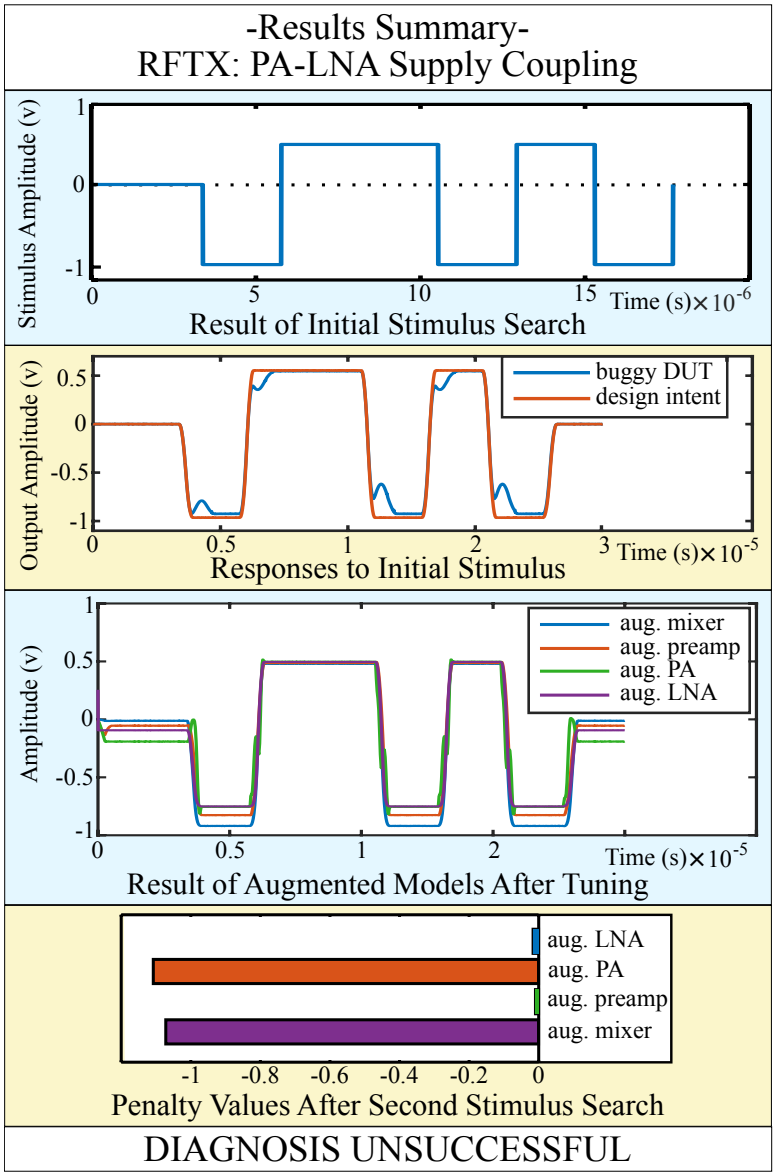

Figure 5.2: Summary of RF Transceiver Experiment Two

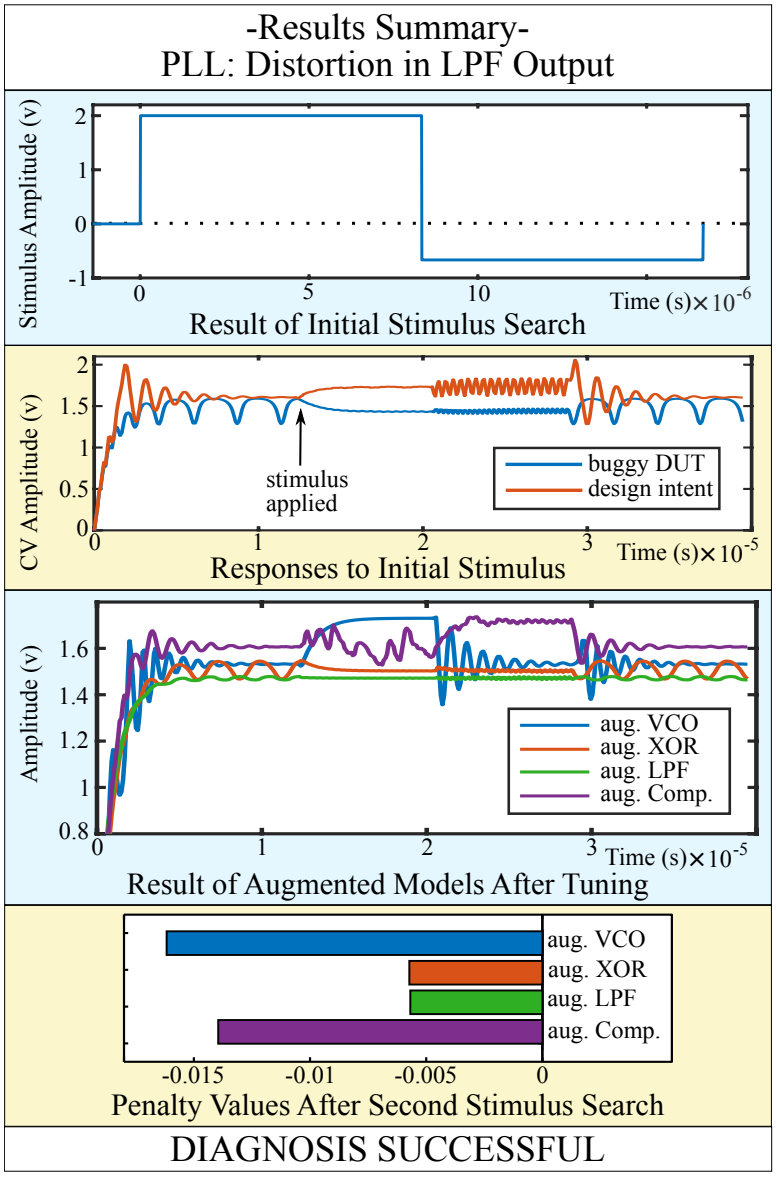

Figure 5.3: Summary of Results for Diagnosis of PLL w/buggy LPF

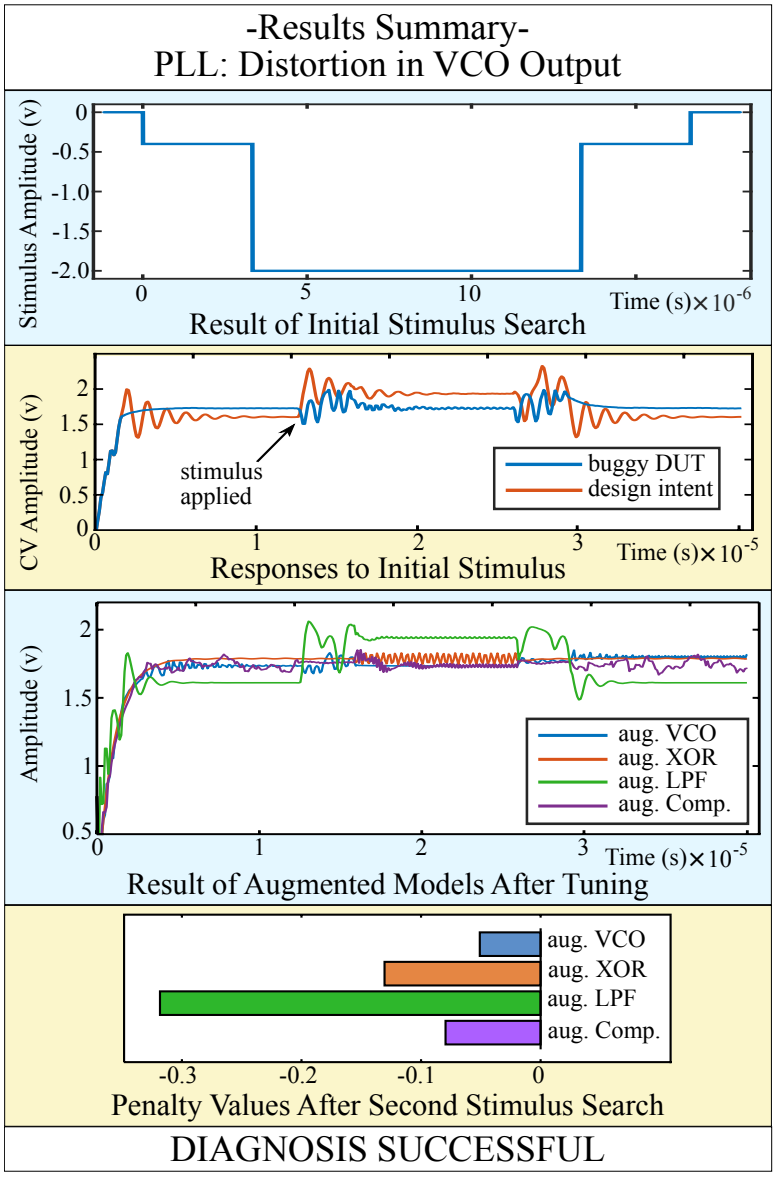

Figure 5.4: Summary of Results for Diagnosis of PLL w/buggy VCO

Appendices

APPENDIX A

GLOSSARY

analog and mixed-signal (AMS) A class of cirucuit/system that includes a mixture of analog, digital, or radio-frequency (RF) componentry.

arbitrary-waveform generator (AWG) A piece of test equipment capable of generating an aritrary high-speed sequence of analog voltages (waveform).

artificial neural network (ANN) A processing paradigm inspired by nervous system biology, an regular structure of highly connected simple units.

bug A localized condition of variation of functional behavior, due to design oversight, model inadequacy, or manufacturing.

built-in self-test (BIST) An auxillary set of features included on-die whose purpose is to test other components.

design-for-test (DFT) A category of design techniques which increase testability of a fabricated system.

device-under-validation (DUV) A system, either fabricated or simulated, which is undergoing design validation.

device-under-test (DUT) An instance of a design, either fabricated or simulated, which is undergoing (a battery of) tests.

digitizer A piece of test equipment capable of recording a high-speed sequence of analog voltages (waveform).

fault A localized condition of variation of an electrical parameter, usually due to manufacturing. Faults can be extreme (a short-circuit), and they can be subtle (an inductance slightly larger than expected).

fault model An abstract description of a plausible manifestation of an electrical fault.

low-noise amplifier (LNA) A radio-frequency amplifier designed to impart minimal noise to its output signal.

low-pass filter (LPF) A filter designed to pass low frequencies and attenuate high frequencies.

manufacturing test *See* production test.

Markov Decision Process (MDP) A conceptual class of system which is a stochastic variant of a state-machine. State transitions are expressed as probability distributions contingent upon the present state of the system and which do not change over time.

maximum likelihood estimate (MLE) The solution to a class of problem wherein the likelihood that a given set of data could be produced by various parameterizations of some probabilistic model. The maximum likelihood estimate are those parameters which are most likely to have explained the data.

mean square error (MSE) A goodness-of-fit metric defined as

$$\frac{1}{n}\sum_{i}(x_{o_i} - x_{p_i})^2$$

where n is the number of observations, x_{o_i} is an individual observed value, and x_{p_i} is an individual predicted value.

model An analytical, behavioral, or numerical description of a particular design element.

model-order reduction (MOR) The study of classes of technique for reducing the degree of mathematical complexity in a system dynamical model. A cannonical goal is to reduce the dimensionality of the state-space or phase-space of a system described by a set of differential equations.

Multivariate Adaptive Regression Splines (MARS) A statistical regression tool introduced by Jerome Friedman in his 1991 paper by the same title.

post-silicon A qualifier used to describe pre-production activities which include fabricated silicon prototype systems.

power amplifier (PA) A radio-frequency amplifier designed to impart maximal power to its output signal.

production test Refers to a battery of tests performed on fabricated systems to ensure all performance specifications are met; systems which pass are shipped to customers, systems which fail are discarded.

radio-frequency (RF) A class of circuits envolved in the transmission and/or reception of electromagnetic signals in the frequency range [3kHz, 10 THz].

regularized least-squares (RLS) The solution to the linear alegraic inverse problem: $Ax = b$ subject to a "regularizing term" which penalizes candidates of A which contain large coefficients.

stimulus *See* test stimulus.

support vector machine (SVM) A type of machine learning classifier.

system-on-chip (SoC) A (typically mixed analog/digital) complete system composed of a diversity of componentry and integrated on one die.

test escape Refers to faulty parts which avoid detection in production testing.

test coverage An efficacy metric for production tests; usually (detectable failure modes) / (possible failure modes).

test stimulus A particular sequence of voltages applied to terminal(s) of a DUT while undergoing a test.

voltage-controlled oscillator (VCO) An oscillator whose frequency of oscillation is determined by the voltage present on an input terminal.

yield loss Refers to fault-free parts which are incorrectly labed faulty and discarded.

APPENDIX B

LIST OF PUBLICATIONS

[1] S. Deyati, B. J. Muldrey, and A. Chatterjee, "Trojan detection in digital systems using current sensing of pulse propagation in logic gates," in *2016 17th International Symposium on Quality Electronic Design (ISQED)*, Mar. 2016, pp. 350–355.

[2] ——, "Targeting hardware trojans in mixed-signal circuits for security," in *2016 IEEE 21st International Mixed-Signal Testing Workshop (IMSTW)*, Jul. 2016, pp. 1–4.

[3] B. Muldrey, S. Deyati, and A. Chatterjee, "Post-Silicon Validation: Automatic Characterization of RF Device Nonidealities Via Iterative Learning Experiments on Hardware," in *VLSI Design Conference*, Hyderabad, India, 2016.

[4] B. Muldrey, S. Deyati, M. Giardino, and A. Chatterjee, "RAVAGE: Post-silicon validation of mixed signal systems using genetic stimulus evolution and model tuning," in *VLSI Test Symposium (VTS), 2013 IEEE 31st*, Apr. 2013, pp. 1–6.

[5] N. Tzou, D. Bhatta, B. J. Muldrey, T. Moon, X. Wang, H. Choi, and A. Chatterjee, "Low Cost Sparse Multiband Signal Characterization Using Asynchronous Multi-Rate Sampling: Algorithms and Hardware," *Journal of Electronic Testing*, vol. 31, no. 1, pp. 85–98, 2015.

[6] S. Deyati, B. J. Muldrey, A. Singh, and A. Chatterjee, "High Resolution Pulse Propagation Driven Trojan Detection in Digital Logic: Optimization Algorithms and Infrastructure," in *Test Symposium (ATS), 2014 IEEE 23rd Asian*, IEEE, 2014, pp. 200–205.

[7] S. Deyati, B. J. Muldrey, and A. Chatterjee, "TRAP: Test Generation Driven Classification of Analog/RF ICs Using Adaptive Probabilistic Clustering Algorithm," in *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, Jan. 2016, pp. 463–468.

[8] S. Deyati, A. Banerjee, B. J. Muldrey, and A. Chatterjee, "VAST: Post-Silicon VAlidation and Diagnosis of RF/Mixed-Signal Circuits Using Signature Tests," in *2013 26th International Conference on VLSI Design and 2013 12th International Conference on Embedded Systems*, Jan. 2013, pp. 314–319.

[9] A. Chatterjee, S. Deyati, and B. J. Muldrey, "Post Silicon Validation of Analog/Mixed Signal/RF Circuits and Systems: Recent Advances," in *2016 IEEE 21st International Mixed-Signal Testing Workshop (IMSTW)*, Jul. 2016, pp. 1–6.

[10] A. Chatterjee, S. Deyati, B. Muldrey, S. Devarakond, and A. Banerjee, "Validation Signature Testing: A Methodology for Post-Silicon Validation of Analog/Mixed-Signal Circuits," in collab. with undefined, ser. International Conference on Computer Aided Design, ACM, 2012, pp. 553–556, ISBN: 978-1-4503-1573-9.

[11] D. Banerjee, B. Muldrey, X. Wang, S. Sen, and A. Chatterjee, "Self-Learning RF Receiver Systems: Process-Aware Real-Time Adaptation to Channel Conditions for Low Power Operation," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 2016.

[12] D. Banerjee, B. Muldrey, S. Sen, X. Wang, and A. Chatterjee, "Self-Learning MIMO-RF Receiver Systems: Process Resilient Real-Time Adaptation to Channel Conditions for Low Power Operation," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2014, pp. 710–717.

[13] B. Muldrey, "Mixed Signal Design Validation Using Reinforcement Learning Guided Stimulus Generation for Behavior Discovery," in *Proceedings IEEE VLSI Test Symposium*, Monterey, CA, USA: IEEE, Apr. 2019.

[14] D. Charalampidis and B. Muldrey, "Clustering using multilayer perceptrons," *Nonlinear Analysis*, vol. 71, no. 12, e2807–e2813, 2009.

[15] B. Muldrey, S. Deyati, and A. Chatterjee, "Concurrent Stimulus and Defect Magnitude Optimization for Detection of Weakest Shorts and Opens in Analog Circuits," in *2016 IEEE 25th Asian Test Symposium (ATS)*, Nov. 2016, pp. 96–101.

[16] S. Deyati, B. J. Muldrey, A. Banerjee, and A. Chatterjee, "Atomic model learning: A machine learning paradigm for post silicon debug of RF/analog circuits," in *2014 IEEE 32nd VLSI Test Symposium (VTS)*, Apr. 2014, pp. 1–6.

[17] S. Deyati, B. J. Muldrey, and A. Chatterjee, "Adaptive testing of analog/RF circuits using hardware extracted FSM models," in *2016 IEEE 34th VLSI Test Symposium (VTS)*, Apr. 2016, pp. 1–6.

[18] B. Muldrey, S. Deyati, and A. Chatterjee, "DE-LOC: Design validation and debugging under limited observation and control, pre- and post-silicon for mixed-signal systems," in *2016 IEEE International Test Conference (ITC)*, Nov. 2016, pp. 1–10.

[19] S. Deyati, B. J. Muldrey, A. D. Singh, and A. Chatterjee, "Challenge Engineering and Design of Analog Push Pull Amplifier Based Physically Unclonable Function for Hardware Security," in *2015 IEEE 24th Asian Test Symposium (ATS)*, Nov. 2015, pp. 127–132.

[20] N. L. Tzou, D. Bhatta, X. Wang, T.-H. Chen, S.-W. Hsiao, B. Muldrey, H. W. Choi, and A. Chatterjee, "Concurrent Multi-Channel Crosstalk Jitter Characterization Using Coprime Period Channel Stimulus," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 63, no. 6, pp. 859–870, 2016.

[21] S. Deyati, A. Chatterjee, and B. J. Muldrey, "Analog Push Pull Amplifier-Based Physically Unclonable Function for Hardware Security," US20170126415A1, May 2017.

[22] S. Deyati, B. Muldrey, and A. Chatterjee, "BISCC: Efficient pre through post silicon validation of mixed-signal/RF systems using b uilt ins tate c onsistency c hecking," in *Proceedings of the Conference on Design, Automation & Test in Europe*, European Design and Automation Association, 2017, pp. 274–277.

[23] S. Deyati, A. Chatterjee, and B. J. Muldrey, "Analog push pull amplifier-based physically unclonable function for hardware security," Feb. 2019.

[24] S. Deyati, B. Muldrey, A. Singh, and A. Chatterjee, "Design of efficient analog physically unclonable functions using alternative test principles," in *2017 International Mixed Signals Testing Workshop (IMSTW)*, IEEE, 2017, pp. 1–4.

[25] N. Tzou, D. Bhatta, B. J. Muldrey Jr, T. Moon, X. Wang, H. Choi, A. Chatterjee, C. S. M. Signal, and C. U. A. Multi-Rate, "2015 JETTA-TTTC Best Paper Award," *J Electron Test*, vol. 32, pp. 659–660, 2016.

[26] S. Deyati, B. J. Muldrey, B. Jung, and A. Chatterjee, "Concurrent built in test and tuning of beamforming MIMO systems using learning assisted performance optimization," in *2017 IEEE International Test Conference (ITC)*, IEEE, 2017, pp. 1–10.

APPENDIX C
SOFTWARE TOOL DEVELOPMENT

C.1 Pyspectre: A Modern Python Interface to Cadence Spectre

C.1.1 Introduction

Pyspectre is the name of a tool I developed in the course of this research in order to provide a lightweight, low system-overhead python interface to the Cadence Spectre® circuit simulator that would allow multiple instances of Spectre to be held in memory, manipulated, evaluated, and re-evaluated with minimal cpu, memory, or disk overhead. Pyspectre utilizes an undocumented "interactive mode" of the Spectre binary which implements a "read, evaluate, print loop" (REPL) and accepts a number of commands through standard input. Pyspectre carefully invokes, maintains, and interacts with a running instance of Spectre inside a subprocess while extending access to other objects in the python environment. Pyspectre is available on the Python Packaging Index for installation; its source code is available on Gitlab.com for use and contribution [102].

C.1.2 Motivation

Modern reinforcement learning algorithms operate on the assumption that their environment can be framed as a Markov decision process. Framed as such, RL agents take action on a periodic basis and they are trained in the same manner. And so, in a circuit simulation environment, this corresponds to periodically pausing the simulation, passing the circuit's state to the RL agent, asking the agent to infer what subsequent action should be taken, and manipulating circuit inputs to reflect the RL agent's choice.

Cadence provides programmatic interface to Spectre through their "Ocean" tool so that scripts can be written in their "Skill" language for automated simulation. These mecha-

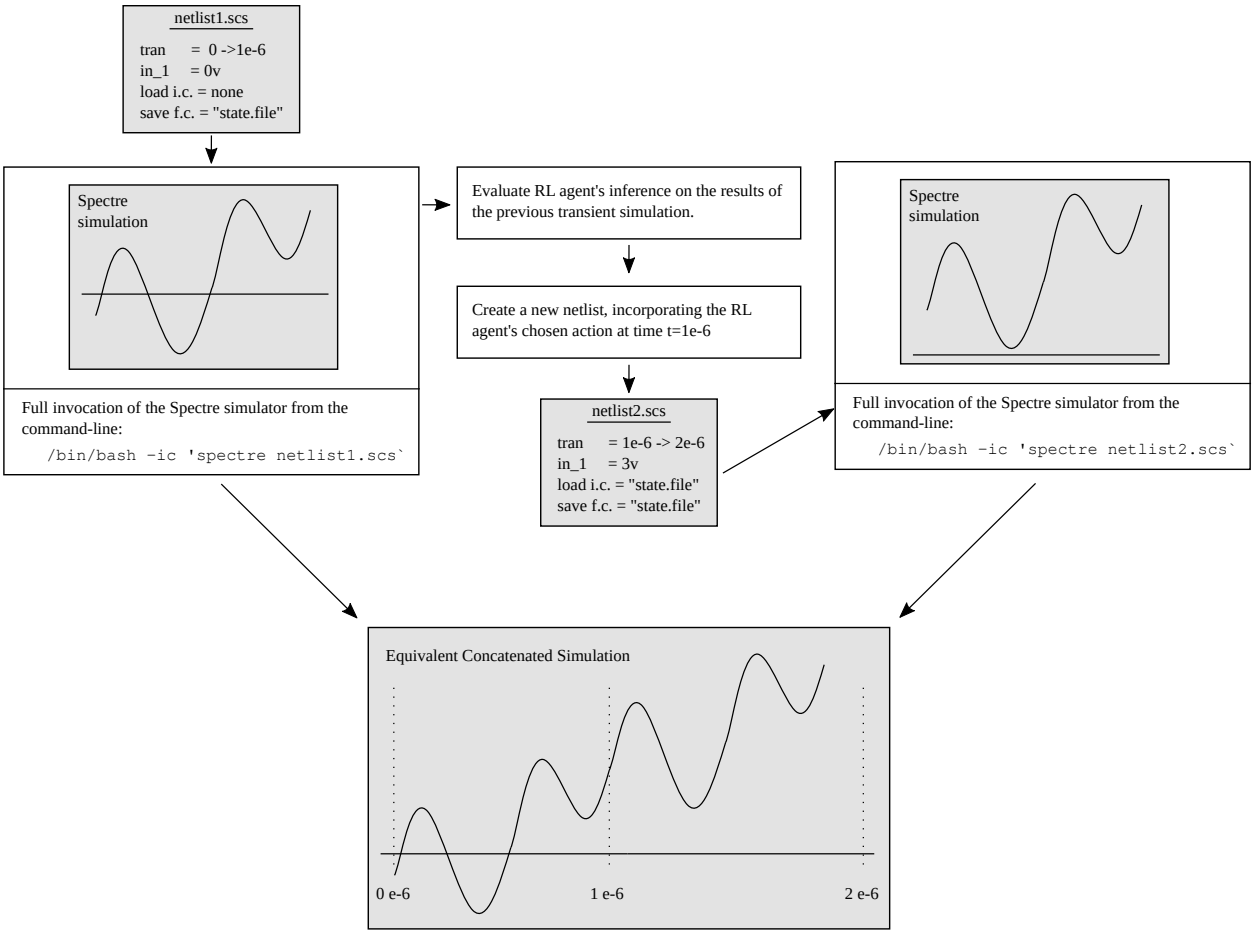

Figure C.1: An illustration of how a reinforcement learning algorithm would interact with Cadence Spectre through Cadence's provided command-line interface.

nisms are not suitable for realtime, on-the-fly manipulation of a stimulus over the course of a single simulation because the Ocean scripts are statically interpreted. Additionally, Cadence provides a command-line interface for Spectre, but one would be limited to creating individual netlists and running individual simulations for each action taken by the RL agent. Figure C.1.2 illustrates that process:

Either of the available mechanisms seemed inadequate considering that a single test stimulus would consist of hundreds of individual RL learner actions. A great deal of time would be wasted having to create new Spectre processes, allocate memory, and parse the (nearly identical) netlist each time.

C.1.3 Design Considerations

The primary design consideration for Pyspectre was speed. In the course of a single training session, a reinforcement learning algorithm may be exposed to tens of thousands of "episodes" (each episode corresponding to one stimulus). With each episode consisting of hundreds of RL actions, it's quite feasible that on the order of millions of individual transient simulations would be conducted during a training session. Pyspectre sought to minimize the computational overhead for each evaluation.

Parallelism

A secondary consideration of Pyspectre was to enable multiple instances of Spectre to be maintained simultaneously. In order to do this, several default behaviors had to be modified to prevent Spectre instances to have write collisions on various files.

Circuit-object Portability

Additional utility in Pyspectre would come from bundling all the requisite simulation collateral including netlists, stimulus files, model files, etc. into a serializeable python object. In this way, simulations could be easily sent across the network for evaluation on remote machines.

Compiled Model Maintenance

Early iterations of Pyspectre created temporary directories in the file system within which to operate individual instances and maintain individual circuit state files. The arrangement overlooked the production of compiled AHDL models which by default were created in these directories. As a result, each instantiation would re-compile AHDL models that may have previously been compiled.

C.1.4 Implementation Details

Pyspectre was implemented in the Python language, in a manner compatible with Python v2.7 and v3.x. The "pexpect" library was used for its "ReplWrapper" class which spawned the spectre process and provided a string interface for interaction.

C.1.5 Libpsf

Libpsf is the name of a legacy project by Henrik Johansson [106] which implements a PSF file reader in the C/C++ language. PSF is the native (proprietary) file format of the Cadence simulators, whose structure, as far as I know, hasn't been released publicly. I presume that Mr. Johansson has done the work of manually parsing and reverse engineering the file format; in any event, he has made his code available for all on Github.

Unfortunately, Libpsf was not a turn-key Python installation like so many others. The Python bindings were well out of date, having been implemented in 2014 using a flavor of the Boost::Python library which was built against legacy versions of the Numpy numerical computation libary [107] and [108]. Libpsf would not compile with the contemporary releases of Numpy which are used by all modern Python tools.

In order to extract data from Spectre in its native and compact binary format, I would have to update the Libpsf codebase to interface with modern Numpy. In order to achieve this, I followed the distribution recommendations of the Python Packaging Authority, using their system images based on legacy CentOS releases to compile a maximally compatible version of Libpsf. The source code was modified for python 2.7/3.6/3.7 compatibility and was linked against Boost v1.68 and Numpy v1.15. The compiled code was then packaged into "wheels" for each of the Python versions and was uploaded for distribution on Pypi.org [103].

	n	Mean	Min	Max	St.D.
Spectre CLI interface	100	811	558	3006	307
Pyspectre interface	100	24	18	58	4.5

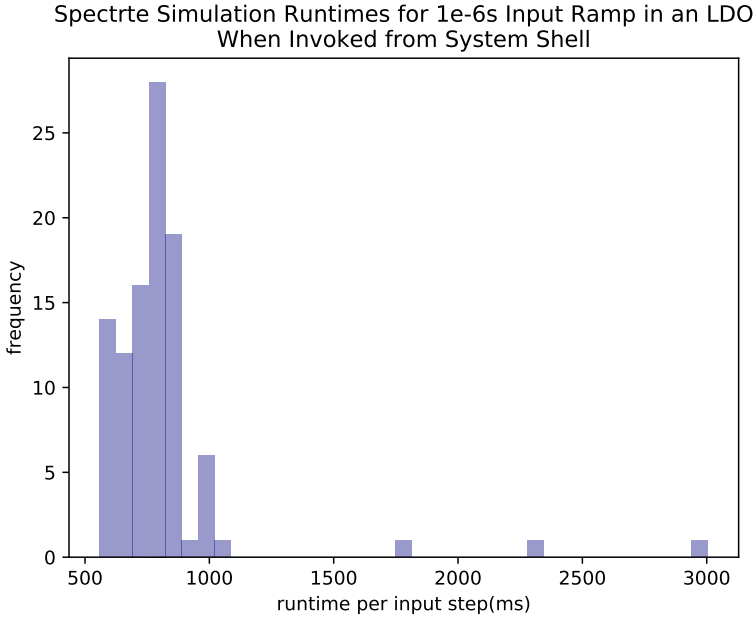

Figure C.2: A histogram of runtimes of input action steps taken in an LDO Spectre simulation when using Cadence's command-line interface.

C.1.6 Performance

To measure performace, I used a netlist which includes a transistor-level low-dropout DC regulator as well as a behavioral-level version of the same circuit. To measure performance, I setup an experiment which will simulate a single reinforcement learner action (a transient simulation from $0s$ to $1\mu s$) and will take that same action 100 times first using Spectre's command-line interface invocation, and 100 times again using the pyspectre interface. We expect that the simulation time of a single step should be relatively small, and that the pyspectre interface will save the cost of netlist read-in at every step.

Figure C.1.6 and Figure C.1.6 show histograms of per-step execution times for the command-line interface and the Pyspectre interface respectively. Table C.1.6 provides the descriptive statistics of the experiment. In this example, an RL algorithm should expect a 34x reduction in simulation time.

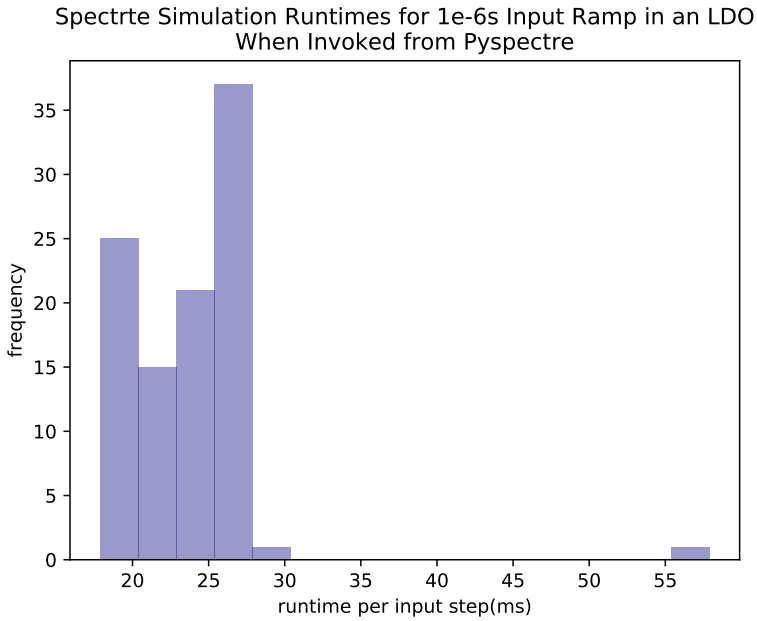

Figure C.3: A histogram of runtimes of input action steps taken in an LDO Spectre simulation when using the Pyspectre interface.

C.1.7 Conclusion

One bottleneck to speed is Spectre's file system activity. To further enhance performance, disk IO should be minimized. Two ways of approaching this immediately would be to: either a) attempt the use of command-line options to disable all logging and simulation results output into the file-system, instead printing all simulation results to STDOUT and having pyspectre interpret them there; or b) have pyspectre create the `${PYSPECTRE_ROOT}/simulation` subdirectory as a ramdisk, so that pyspectre could maintain its contents entirely in system memory with minimal change elsewhere in the code.

C.2 Circuitgym: Extending OpenAI Baselines to Circuits

C.2.1 Introduction

Circuitgym is the name of a python library I created to meet the requirement of creating a Markov Decision Process (MDP)-style interface compatible with the OpenAI "Gym"

API for circuit simulations. Such an interface was required for conducting reinforcement learning experiments in a circuit environment.

C.2.2 Motivation

In the course of my research, the question arose whether there were better means of re-using simulation results carried out by previous optimization routines. Reinforcement learning showed promise in this regard; at any point in time, an AI actor could be trained using a collected history of data. Initial experimentation was done by implementing a few popular RL algorithms in Matlab and experimenting on systems of Volterra filters (Chapter 3). I found that my algorithms were not parametrically robust, and so I turned to the more rigorously tested implementations found in the open-source library, "Baselines," released by the OpenAI research company [97]. The reparameterizations of the algorithms as implemented by Baselines proved easier to use and more stable than my own crude Matlab implementations. It was when I sought to replace the Volterra filter models with spice circuit simulations that I realized an additional tool was needed to form a layer between the spice simulation back-end and the reinforcement learning agent.

C.2.3 Design Considerations

Many of the major reinforcement learning libraries, including not only OpenAI Baselines but also Keras-RL [109] and Stable-Baselines [98], interact with the underlying environment via the same API, defined by Baselines. The API is fairly simple: the environment must respond to a "reset" command, it must implement a quantified "action space" from which an agent can select actions, and after every action, it should return a vector of new observations reflecting changes in the environment transpiring as a result of the previous action.

With the API spelled out, I needed only to wrap a spice simulation in a way that a sequence of discrete actions could be taken, with the spice simulation progressing in the

background in such a way that agent actions were accounted for in the simulation. Because latter portions of the simulation required input and state information which are not yet known, simulations cannot be carried out in their entirety as single spice analyses. Instead, an individual spice simulation would have to be run for each step of the reinforcement learning "episode." For this reason, Pyspectre was developed Section C.1 . But additionally, Circuitgym required standardization of the circuit input mechanism, a way to avoid simulation discontinuities between adjacent simulations, and a configurable way to return information about the intervening simulation.

C.3 Xanity: An Experiment Runner and Data Management Tool

C.3.1 Introduction

"Xanity" is the name of a software tool which came gradually into existence over the course of my research. Every time I'd embark on a new course of experimentation, I would bring certain pieces of code with me, in order to facilitate the logistical side of the work: source code snap-shotting and version control, notes on individual trials, dependencies on external libraries, etc. Eventually, Xanity was given a name and became a project in its own right.

C.3.2 Motivation

The two central complications and nuisances in my research which Xanity addresses arise from the use of virtual environments for individual lines of experimentation and from the need for book-keeping in the volumes of data produced during debugging and bona fide experimental runs.

Virtual environments created by tools like "virtualenv," "venv," "pipenv," and Anaconda are very useful and popular for managing the webs of dependencies created when using mixtures of external tools. Individual circumstances have particular version requirements and system library requirements; additionally, portability and collaboration require delegating and codifying the task of managing code environments. As a result, rather than

managing libraries, experimentalists now have the task of managing environments. Any individual experiment must be run from inside the correct environment, and the constant context switching can become cumbersome.

Additionally, experiments generate data; both in bona fide experimentation and during experiment debugging as well. Some data has a global scope and is recalled by many experiments, and other data is regenerated routinely with every run. Some data is computationally costly, and should be reused, even if only for debugging purposes, and other data is cheap, kept only for documentation or record-keeping. Without a book-keeping mechanism, keeping track of the various kinds and pieces of data can be quite burdensome, tedious, and costly in terms of disk space.

The major design goals of Xanity were as follows:

1. **Collocation of Experiment Collateral** In order for projects to be more portable (both within one system and across systems), all dependencies in terms of source code, environment, and data should all be collocated. Should an experimentalist have to move project directories across file systems, the integrity and usability of the experimental setup should not be compromised.

2. **Clear Cataloging of Data** To mitigate the burden of keeping track of various ad-hoc data-storage regimes, Xanity should implement a self-explanatory and generalizable means of organizing and storing data.

3. **Simplification of Setup** The process of getting a new system ready for experimentation can be daunting. Both Anaconda and Pip have mechanisms for snap-shotting the states of environments which enable easy replication. Use of these mechanisms often is not straightforward and requires both the collaborator and collaborate-ee to be familiar with the advanced usage of the tools.

4. **Repeatability** Critical to the progression of exploration itself, repeatability should extend from copying a project or base of code all the way through production of the

concluding analysis. The complexity in modern tool chains and setups represents tremendous variability which, left un-addressed, can compromise the repeatability of experiments.

C.3.3 Implementation

Central to the implementation of Xanity is the desire to leverage existing tools from the software development world like Git revision control tools and Anaconda and Pip package managers, but in such a way that a scientific experimentalist could benefit from them without having to become expert in their use.

For environment management, Xanity can delegate environment creation and entrance/exit to Anaconda and Pip. Xanity must, however, exist both inside and outside the experimental environments. For running experiments, execution control must initiate in the system environment where Xanity can orient itself, parse user requests, and then "step into" the appropriate environment. It can then pass control off to the Xanity instance inside. In order to do this, Xanity must make sure that Xanity itself is installed whenever it creates new environments. A repeatability problem arises, however if Pip is used; there could be a change in Xanity version between Xanity installation at the system level and Xanity installation during environment creation. For this reason, Xanity is designed to be independently *self-replicating*. During the initial installation at the system level, Xanity creates a symbolic copy of itself as an installable python package which it will use to install Xanity into experiment environments.

While there are databases and tools abounding for data management, none are quite as portable, transparent, and user-friendly as the file system itself. Use of the file system enables other tools like Git and other human users to easily parse and keep track of data in a familiar way. And so, Xanity implements a data directory structure that is easy to understand and easy for Xanity to parse and recall experimental data.

For recalling data, Xanity implements a data-query API in which all matching data can

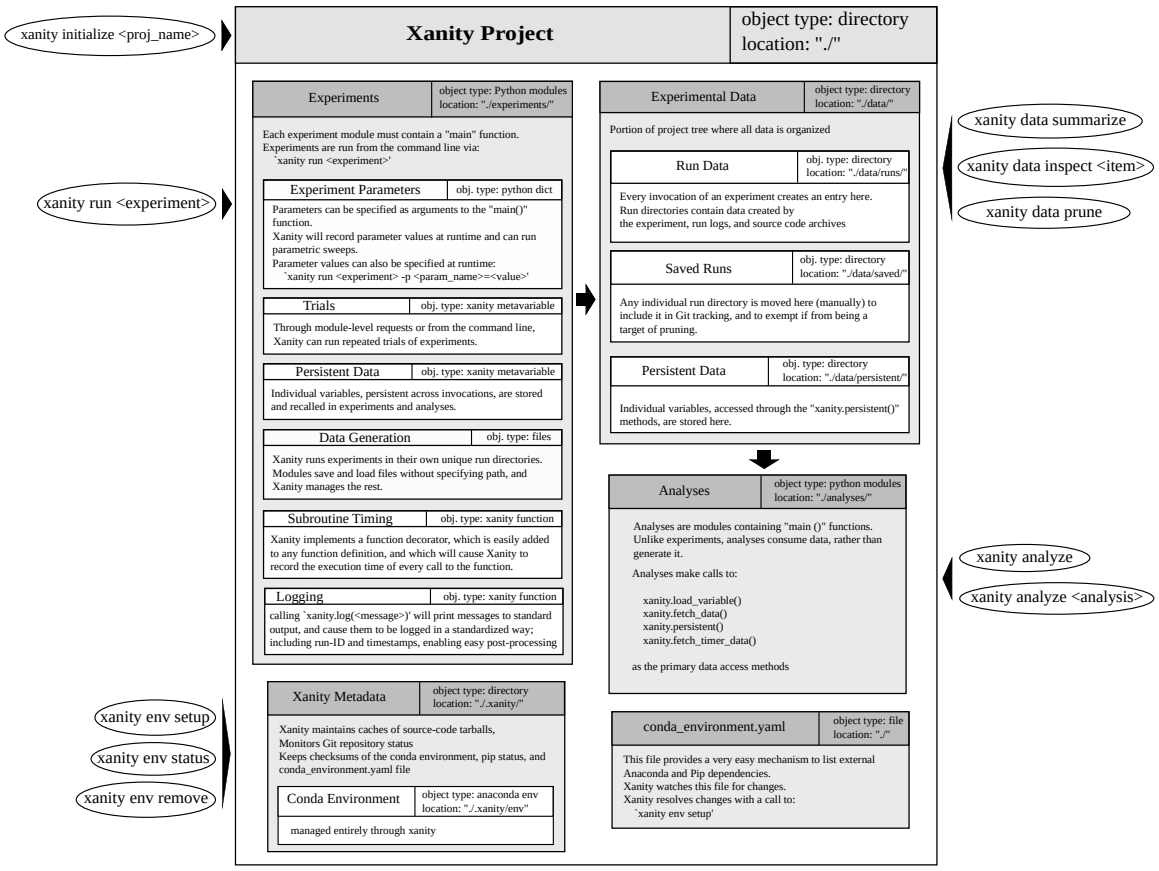

Figure C.4: A graphical representation of the Xanity framework, illustrating commandline entrypoints for interacting with a Xanity project.

be easily recalled by any experiment or analysis in a Pandas DataFrame object containing experiment metadata for each match. A limited number of data management tools have been implemented as well. Xanity has components for summarizing the data associated with a project and associated with individual experiments within. Xanity also has mechanisms for scoring the "value" of data saved in an individual run so that empty and low-value run directories can be identified and pruned.

C.4 Waverunner: A Lightweight Parallelizing RPC Server

C.4.1 Introduction

Waverunner is the name of a remote job execution tool developed in the course of this research. It was developed to address the specific problems faced in our lab environment: having to run simulations on computational workers which cannot be addressed directly on the network; and having to run simulations in batches, with the worker remaining online to accept new batches of simulation. The tool is available on the Python Packaging Index for installation; its source code is available on Gitlab.com for use and contribution [110].

C.4.2 Motivation

Early in the course of conducting experiments it became apparent that the computational resources of a desktop workstation would be insufficient to run the large number of simulations required to enable some of the methods being explored. We had developed pyspectre to enable very rapid parallel instantiation and manipulation of multiple Cadence Spectre® instances. We were in search of a mechanism to offload simulation-computation duties to more pyspectre instances than a single workstation can run. Georgia Tech operates a shared cluster computing environment known as "PACE" for this purpose exactly. It turns out, however, that the off-the-shelf solution to working with PACE was not a good fit.

Our iterative algorithms required the evaluation of on the order of 10,000 simulations per batch. This quantity (modest circuits) could be simulated on 8 cores using parallelized pyspectre in on the order of 10 minutes. Unfortunately, the job queues for receiving ˜8 cores from PACE were often on the order of 3-5 minutes.

For optimal resource allocation, task queuing time should be negligible relative to the work time. In the off-the-shelf usage model, the queuing-to-work fraction was about 30%, far from negligible. In our case, the computing resource should be used for at least 300-500 minutes to justify its own wait time. This would mean that we should either increase batch

size by 30-50x or reuse the same resource multiple times after receiving it from PACE. The first option was unattractive because statistical analysis suggested that increasing the batch size was unlikely to result in proportional returns in terms of new information. The second option was attractive because it would potentially allow additional computational resources to remain on standby, our validation algorithms able to dynamically offload batches or parts of batches for external simulation in an on-the-fly fashion.

C.4.3 Design Considerations

The very first exploration was to see what means of communication with the PACE worker nodes were available. If an interactive shell session could be established, Pyspectre could potentially install itself on the worker, send the circuit collateral, and interact with the remote Pyspectre via STDIO. Unfortunately, implementing a mechanism to interface with a remote pyspectre instance through remote shell would be nontrivial. Alternatively, if a port on the remote machine could be opened, a tool could be used for simpler data transmission; however, the PACE network is configured such that worker nodes aren't exposed to the campus-wide network and port forwarding from the login nodes is disabled.

Because using a shell interface was infeasible and direct network access was not available, a PACE worker machine would have to be configured to pull jobs from a job server running outside of the PACE network.

C.4.4 Existing Solutions

The open-source software universe abounds in task-queuing and job submission systems, multiprocess communications systems, and cluster management tools. Several immediately stood out as candidates: Celery, RabbitMQ, Pyro, and Twised Matrix all exist as fairly mature solutions in wide deployment with native Python interfaces. All of them, however, operated in a job "push" paradigm, whereby the central queue system pushed jobs to workers. Additionally, these kinds of tools in general assume that the worker and

queue manager are on the same network and that they have full control over port accessibility, firewall rules, etc.. This was not the case with PACE workers due to the networking limitations mentioned in the previous section.

All official Python distributions include an XML-RPC library implementing a basic client and server. XML-RPC stands for "extensible markup language remote procedure call." The library is intended to allow a client to request a server to execute any function it is configured to serve via simple HTTP GET requests with an XML payload. There is nothing which prevents the library being utilized in a symmetrical fashion, allowing machines on either end of a communications channel to act as both client and server.

C.4.5 Implementation Details

Being constrained primarily by the inability to send TCP packets to PACE workers, the Waverunner server implements 'list_job()', 'get_job()', and 'put_job()' methods, allowing a *worker* to initiate all transactions. Therefore, a Waverunner worker only has to be pointed to the network address and port of another Waverunner server, and it can view a list of jobs to-be-run, can take jobs off the cue, and then submit its results all via HTTP PUT requests that it initiates.

Implementing workers and queues in this fashion also allows for both worker and queuer to exist on the same machine. Thus Waverunner can also function as a mechanism for managing local parallelism.

REFERENCES

[1] J. Keshava, N. Hakim, and C. Prudvi, "Post-silicon validation challenges: How EDA and academia can help," in *47th ACM/IEEE Design Automation Conference (DAC)*, Jun. 2010, pp. 3–7.

[2] X. Li, F. Wang, S. Sun, and C. Gu, "Bayesian Model Fusion: A statistical framework for efficient pre-silicon validation and post-silicon tuning of complex analog and mixed-signal circuits," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2013, pp. 795–802.

[3] L. Yin, Y. Deng, and P. Li, "Simulation-assisted formal verification of nonlinear mixed-signal circuits with Bayesian inference guidance," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 7, pp. 977–990, 2013.

[4] K. Poolla, P. Khargonekar, A. Tikku, J. Krause, and K. Nagpal, "A time-domain approach to model validation," *IEEE Transactions on Automatic Control*, vol. 39, no. 5, pp. 951–959, May 1994.

[5] G. G. E. Gielen and R. A. Rutenbar, "Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits," *PROCEEDINGS OF THE IEEE*, vol. 88, no. 12, p. 28, 2000.

[6] A. Ismail, S. Ibrahim, and M. Dessouky, "An 8Gbps discrete time linear equalizer in 40nm CMOS technology," in *2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2015, pp. 1–4.

[7] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, Mar. 1951.

[8] C. V. deSauty, "Duplex system of telegraphy on submarine cables," *Journal of the Society of Telegraph Engineers*, vol. 2, no. 4, pp. 138–145, 1873.

[9] L. P. Hunter and H. Fleisher, "Graphical Analysis of Some Transistor Switching Circuits," *Proceedings of the IRE*, vol. 40, no. 11, pp. 1559–1562, Nov. 1952.

[10] E. W. Marchant and A. C. Robb, "Methods of testing small servo mechanisms and data-transmission systems," *Journal of the Institution of Electrical Engineers - Part IIA: Automatic Regulators and Servo Mechanisms*, vol. 94, no. 2, pp. 292–297, May 1947.

[11] R. D. Eldred, “Test Routines Based on Symbolic Logical Statements,” *J. ACM*, vol. 6, no. 1, pp. 33–37, Jan. 1959.

[12] W. G. Bouricius, E. P. Hsieh, G. R. Putzolu, J. P. Roth, P. R. Schneider, and C. Tan, “Algorithms for Detection of Faults in Logic Circuits,” *IEEE Transactions on Computers*, vol. C-20, no. 11, pp. 1258–1264, Nov. 1971.

[13] J. P. Roth, W. G. Bouricius, and P. R. Schneider, “Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits,” *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 5, pp. 567–580, Oct. 1967.

[14] S. G. Chappell, “Lamp: Automatic test generation for asynchronous digital circuits,” *The Bell System Technical Journal*, vol. 53, no. 8, pp. 1477–1503, Oct. 1974.

[15] P. Goel, “An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits,” *IEEE Transactions on Computers*, vol. C-30, no. 3, pp. 215–222, Mar. 1981.

[16] P. O. Farnham and A. W. Barber, “Problems Involved in the Design and Use of Apparatus for Testing Radio Receivers,” *Proceedings of the Institute of Radio Engineers*, vol. 18, no. 8, pp. 1338–1350, Aug. 1930.

[17] E. Babakrpur and W. Namgoong, “A Dual-Path 4-Phase Nonuniform Wideband Receiver With Digital MMSE Harmonic Rejection Equalizer,” *IEEE Transactions on Microwave Theory and Techniques*, vol. PP, no. 99, pp. 1–10, 2017.

[18] F. Azais, “Analog test: Why still a la mode after more than 25 years of research?” In *2015 20th IEEE European Test Symposium (ETS)*, May 2015, pp. 1–1.

[19] S. Sunter, K. Jurga, P. Dingenen, and R. Vanhooren, “Practical random sampling of potential defects for analog fault simulation,” in *Test Conference (ITC), 2014 IEEE International*, Oct. 2014, pp. 1–10.

[20] M. Aminian and F. Aminian, “Neural-network based analog-circuit fault diagnosis using wavelet transform as preprocessor,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 2, pp. 151–156, Feb. 2000.

[21] M. Slamani and B. Kaminska, “Multifrequency analysis of faults in analog circuits,” *IEEE Design Test of Computers*, vol. 12, no. 2, pp. 70–80, Sum. 1995.

[22] J. W. Bandler and A. E. Salama, “Fault diagnosis of analog circuits,” *Proceedings of the IEEE*, vol. 73, no. 8, pp. 1279–1325, Aug. 1985.

[23] M. Tadeusiewicz, S. Halgas, and M. Korzybski, "An algorithm for soft-fault diagnosis of linear and nonlinear circuits," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, no. 11, pp. 1648–1653, Nov. 2002.

[24] S. Cherubal and A. Chatterjee, "Test generation based diagnosis of device parameters for analog circuits," W. Nebel and A. Jerraya, Eds., IEEE Comput. Soc, 2001, pp. 596–602, ISBN: 978-0-7695-0993-8.

[25] N. Sen and R. Saeks, "Fault diagnosis for linear systems via multifrequency measurements," *IEEE Transactions on Circuits and Systems*, vol. 26, no. 7, pp. 457–465, Jul. 1979.

[26] M. Slamani and B. Kaminska, "Analog circuit fault diagnosis based on sensitivity computation and functional testing," *IEEE Design Test of Computers*, vol. 9, no. 1, pp. 30–39, Mar. 1992.

[27] A. Banerjee and A. Chatterjee, "Automatic Test Stimulus Generation for Diagnosis of RF Transceivers Using Model Parameter Estimation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 12, pp. 3114–3118, Dec. 2015.

[28] J. W. Jeong, J. Kitchen, and S. Ozev, "Robust amplitude measurement for RF BIST applications," IEEE, May 2015, pp. 1–6, ISBN: 978-1-4799-7603-4.

[29] E. S. Erdogan and S. Ozev, "Detailed Characterization of Transceiver Parameters Through Loop-Back-Based BiST," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 6, pp. 901–911, Jun. 2010.

[30] H.-G. Stratigopoulos and S. Sunter, "Efficient Monte Carlo-based analog parametric fault modelling," in *VLSI Test Symposium (VTS), 2014 IEEE 32nd*, Apr. 2014, pp. 1–6.

[31] ——, "Fast Monte Carlo-Based Estimation of Analog Parametric Test Metrics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 12, pp. 1977–1990, Dec. 2014.

[32] M. Catelani and A. Fort, "Soft fault detection and isolation in analog circuits: Some results and a comparison between a fuzzy approach and radial basis function networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 51, no. 2, pp. 196–202, Apr. 2002.

[33] R. Spina and S. Upadhyaya, "Linear circuit fault diagnosis using neuromorphic analyzers," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, no. 3, pp. 188–196, Mar. 1997.

[34] M. Abu El-Yazeed, "An integrated approach for analog circuit testing using autocorrelation analysis, singular value decomposition and probabilistic neural network," in *Proceedings of the 15th International Conference on Microelectronics, 2003. ICM 2003*, Dec. 2003, pp. 41–45.

[35] S. Chakrabarti and A. Chatterjee, "Fault modeling and fault sampling for isolating faults in analog and mixed-signal circuits," in *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, 1999. ISCAS '99*, vol. 2, Jul. 1999, 444–447 vol.2.

[36] P. Variyam, S. Cherubal, and A. Chatterjee, "Prediction of analog performance parameters using fast transient testing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 3, pp. 349–361, Mar. 2002.

[37] B. Muldrey, S. Deyati, and A. Chatterjee, "DE-LOC: Design validation and debugging under limited observation and control, pre- and post-silicon for mixed-signal systems," in *2016 IEEE International Test Conference (ITC)*, Nov. 2016, pp. 1–10.

[38] N. Nagi, A. Chatterjee, and J. A. Abraham, "A signature analyzer for analog and mixed-signal circuits," in *Proceedings 1994 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, Oct. 1994, pp. 284–287.

[39] L. Milor and V. Visvanathan, "Detection of catastrophic faults in analog integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 2, pp. 114–130, Feb. 1989.

[40] P. Variyam and A. Chatterjee, "Specification-driven test design for analog circuits," in *, 1998 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 1998. Proceedings*, Nov. 1998, pp. 335–340.

[41] C. Zhang, G. He, and S. Liang, "PCA-Based Analog Fault Detection by Combining Features of Time Domain and Spectrum," in *International Workshop on Intelligent Systems and Applications, 2009. ISA 2009*, May 2009, pp. 1–4.

[42] C. Zhang, S. Liang, G. He, and X. Yan, "A wavelet packets and PCA based method for testing of analog circuits," in *2009 IEEE AUTOTESTCON*, Sep. 2009, pp. 348–352.

[43] R. Voorakaranam, S. Cherubal, and A. Chatterjee, "A signature test framework for rapid production testing of RF circuits," in *Automation and Test in Europe Conference and Exhibition Proceedings 2002 Design*, 2002, pp. 186–191.

[44] S. S. Akbay, A. Halder, A. Chatterjee, and D. Keezer, "Low-cost test of embedded RF/analog/mixed-signal circuits in SOPs," *IEEE Transactions on Advanced Packaging*, vol. 27, no. 2, pp. 352–363, May 2004.

[45] R. Senguttuvan and A. Chatterjee, "Alternate Diagnostic Testing and Compensation of RF Transmitter Performance Using Response Detection," in *25th IEEE VLSI Test Symposium (VTS'07)*, May 2007, pp. 395–400.

[46] A. Banerjee, V. Natarajan, S. Sen, A. Chatterjee, G. Srinivasan, and S. Bhattacharya, "Optimized Multitone Test Stimulus Driven Diagnosis of RF Transceivers Using Model Parameter Estimation," in *2011 24th International Conference on VLSI Design (VLSI Design)*, Jan. 2011, pp. 274–279.

[47] S. S. Akbay and A. Chatterjee, "Built-in test of RF components using mapped feature extraction sensors," in *23rd IEEE VLSI Test Symposium (VTS'05)*, May 2005, pp. 243–248.

[48] A. Banerjee, S. Sen, S. Devarakond, and A. Chatterjee, "Automatic Test Stimulus Generation for Accurate Diagnosis of RF Systems Using Transient Response Signatures," in *VLSI Test Symposium (VTS), 2011 IEEE 29th*, May 2011, pp. 58–63.

[49] S. Chakrabarti and A. Chatterjee, "Partial simulation-driven ATPG for detection and diagnosis of faults in analog circuits," in *IEEE/ACM International Conference on Computer Aided Design, 2000. ICCAD-2000*, Nov. 2000, pp. 562–567.

[50] S. Kook, A. Banerjee, and A. Chatterjee, "Signature Testing and Diagnosis of High Precision S? ADC Dynamic Specifications Using Model Parameter Estimation," in *Test Symposium (ETS), 2011 16th IEEE European*, May 2011, pp. 33–38.

[51] S. Mitra, S. Seshia, and N. Nicolici, "Post-silicon validation opportunities, challenges and recent advances," in *2010 47th ACM/IEEE Design Automation Conference (DAC)*, Jun. 2010, pp. 12–17.

[52] C. Gu, "Challenges in Post-silicon Validation of High-speed I/O Links," in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '12, New York, NY, USA: ACM, 2012, pp. 547–550, ISBN: 978-1-4503-1573-9.

[53] A. Jas, B. Pouya, and N. A. Touba, "Virtual scan chains: A means for reducing scan length in cores," in *Proceedings 18th IEEE VLSI Test Symposium*, 2000, pp. 73–78.

[54] J. Sastry, V. Prakash, and L. Reddy, "A Formal Framework for Verification and Validation of External Behavioral Models of Embedded Systems through Use Case Models," in *4th International Conference on Embedded and Multimedia Computing, 2009. EM-Com 2009*, Dec. 2009, pp. 1–8.

[55] L. Liu, D. Sheridan, W. Tuohy, and S. Vasudevan, "Towards coverage closure: Using GoldMine assertions for generating design validation stimulus," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, Mar. 2011, pp. 1–6.

[56] E. Singerman, Y. Abarbanel, and S. Baartmans, "Transaction based pre-to-post silicon validation," in *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, Jun. 2011, pp. 564–568.

[57] S. Edwards, L. Lavagno, E. A. Lee, and A. Sangiovanni-Vincentelli, "Design of embedded systems: Formal models, validation, and synthesis," *Proceedings of the IEEE*, vol. 85, no. 3, pp. 366–390, Mar. 1997.

[58] B. Muldrey, S. Deyati, M. Giardino, and A. Chatterjee, "RAVAGE: Post-silicon validation of mixed signal systems using genetic stimulus evolution and model tuning," in *VLSI Test Symposium (VTS), 2013 IEEE 31st*, Apr. 2013, pp. 1–6.

[59] S. Deyati, A. Banerjee, B. J. Muldrey, and A. Chatterjee, "VAST: Post-Silicon VAlidation and Diagnosis of RF/Mixed-Signal Circuits Using Signature Tests," in *2013 26th International Conference on VLSI Design and 2013 12th International Conference on Embedded Systems*, Jan. 2013, pp. 314–319.

[60] B. Muldrey, S. Deyati, and A. Chatterjee, "Post-Silicon Validation: Automatic Characterization of RF Device Nonidealities Via Iterative Learning Experiments on Hardware," presented at the VLSI Design Conference, Hyderabad, India, 2016.

[61] D. B. Armstrong, "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets," *IEEE Transactions on Electronic Computers*, vol. EC-15, no. 1, pp. 66–73, Feb. 1966.

[62] P. Kabisatpathy, A. Barua, and S. Sinha, "A pseudo-random testing scheme for analog integrated circuits using artificial neural network model-based observers," in *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002*, vol. 2, Aug. 2002, pp. 465–468.

[63] W. Zhang, X. Li, F. Liu, E. Acar, R. Rutenbar, and R. Blanton, "Virtual Probe: A Statistical Framework for Low-Cost Silicon Characterization of Nanoscale Integrated Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 12, pp. 1814–1827, Dec. 2011.

[64] N. Nagi, A. Chatterjee, A. Balivada, and J. Abraham, "Fault-based automatic test generator for linear analog circuits," in *, 1993 IEEE/ACM International Conference on Computer-Aided Design, 1993. ICCAD-93. Digest of Technical Papers*, Nov. 1993, pp. 88–91.

[65] Z. Guo and J. Savir, "Observer-based test of analog linear time-invariant circuits," in *The First IEEE International Workshop on Electronic Design, Test and Applications, 2002. Proceedings*, 2002, pp. 13–17.

[66] M. Hu, H. Wang, G. Hu, and S. Yang, "Soft fault diagnosis for analog circuits based on slope fault feature and BP neural networks," *Tsinghua Science and Technology*, vol. 12, no. S1, pp. 26–31, Jul. 2007.

[67] H. Yoon, J. Hou, A. Chatterjee, and M. Swaminathan, "Fault detection and automated fault diagnosis for embedded integrated electrical passives," in *International Conference on Computer Design: VLSI in Computers and Processors, 1998. ICCD '98. Proceedings*, Oct. 1998, pp. 588–593.

[68] M. Slamani and B. Kaminska, "Testing analog circuits by sensitivity computation," in *, [3rd] European Conference on Design Automation, 1992. Proceedings*, Mar. 1992, pp. 532–537.

[69] S. Sunter and N. Nagi, "Test metrics for analog parametric faults," in *17th IEEE VLSI Test Symposium, 1999. Proceedings*, 1999, pp. 226–234.

[70] K. Kundert and H. Chang, "Verification of Complex Analog Integrated Circuits," in *IEEE Custom Integrated Circuits Conference, 2006. CICC '06*, Sep. 2006, pp. 177–184.

[71] D. Vasilyev, "A TBR-based Trajectory Piecewise-Linear Algorithm for Generating Accurate Low-order Models for Nonlinear Analog Circuits and MEMS," p. 6,

[72] M. Rewienski and J. White, "A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 2, pp. 155–170, Feb. 2003.

[73] Peng Li and L. Pileggi, "Compact reduced-order modeling of weakly nonlinear analog and RF circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 2, pp. 184–203, Feb. 2005.

[74] J. Roychowdhury, "Reduced-order modeling of time-varying systems," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 10, pp. 1273–1288, Oct. 1999.

[75] B. Bond and L. Daniel, "Stable Reduced Models for Nonlinear Descriptor Systems Through Piecewise-Linear Approximation and Projection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1467–1480, Oct. 2009.

[76] Chenjie Gu and J. Roychowdhury, "Model reduction via projection onto nonlinear manifolds, with applications to analog circuits and biochemical systems," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, USA: IEEE, Nov. 2008, pp. 85–92, ISBN: 978-1-4244-2819-9.

[77] C. Borchers, L. Hedrich, and E. Barke, "Equation-based behavioral model generation for nonlinear analog circuits," in *33rd Design Automation Conference Proceedings, 1996*, Las Vegas, NV, USA: ACM, 1996, pp. 236–239, ISBN: 978-0-89791-779-7.

[78] L. Nathke, V. Burkhay, L. Hedrich, and E. Barke, "Hierarchical automatic behavioral model generation of nonlinear analog circuits based on nonlinear symbolic techniques," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, Paris, France: IEEE Comput. Soc, 2004, pp. 442–447, ISBN: 978-0-7695-2085-8.

[79] A. Karthik, S. Ray, P. Nuzzo, A. Mishchenko, R. Brayton, and J. Roychowdhury, "ABCD-NL: Approximating Continuous non-linear dynamical systems using purely Boolean models for analog/mixed-signal verification," in *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, Jan. 2014, pp. 250–255.

[80] B. N. Bond, Z. Mahmood, Y. Li, R. Sredojevic, A. Megretski, V. Stojanovi, Y. Avniel, and L. Daniel, "Compact Modeling of Nonlinear Analog Circuits Using System Identification via Semidefinite Programming and Incremental Stability Certification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 8, pp. 1149–1162, Aug. 2010.

[81] Y.-C. Wang, A. Komuravelli, P. Zuliani, and E. M. Clarke, "Analog Circuit Verification by Statistical Model Checking," in *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '11, Piscataway, NJ, USA: IEEE Press, 2011, pp. 1–6, ISBN: 978-1-4244-7516-2.

[82] B. Muldrey, S. Deyati, and A. Chatterjee, "Concurrent Stimulus and Defect Magnitude Optimization for Detection of Weakest Shorts and Opens in Analog Circuits," in *2016 IEEE 25th Asian Test Symposium (ATS)*, Nov. 2016, pp. 96–101.

[83] B. Kaminska, K. Arabi, I. Bell, P. Goteti, J. Huertas, B. Kim, A. Rueda, and M. Soma, "Analog and mixed-signal benchmark circuits-first release," in *Test Conference, 1997. Proceedings., International*, Nov. 1997, pp. 183–190.

[84] A. DeOrio, Q. Li, M. Burgess, and V. Bertacco, "Machine Learning-based Anomaly Detection for Post-silicon Bug Diagnosis," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '13, San Jose, CA, USA: EDA Consortium, 2013, pp. 491–496, ISBN: 978-1-4503-2153-2.

[85] S. Ahmadyan, J. Kumar, and S. Vasudevan, "Goal-oriented stimulus generation for analog circuits," in *2012 49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, Jun. 2012, pp. 1018–1023.

[86] S. N. Ahmadyan, J. A. Kumar, and S. Vasudevan, "Runtime verification of nonlinear analog circuits using incremental time-augmented RRT algorithm," in *Proceedings of the Conference on Design, Automation and Test in Europe*, EDA Consortium, 2013, pp. 21–26.

[87] S. Steinhorst and L. Hedrich, "Improving verification coverage of analog circuit blocks by state space-guided transient simulation," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2010, pp. 645–648.

[88] A. Singh and P. Li, "On Behavioral Model Equivalence Checking for Large Analog/Mixed Signal Systems," in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '10, Piscataway, NJ, USA: IEEE Press, 2010, pp. 55–61, ISBN: 978-1-4244-8192-7.

[89] H. Hu, Q. Zheng, Y. Wang, and P. Li, "HFMV: Hybridizing formal methods and machine learning for verification of analog and mixed-signal circuits," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, IEEE, 2018, pp. 1–6.

[90] L. Yin, Y. Deng, and P. Li, "Verifying dynamic properties of nonlinear mixed-signal circuits via efficient SMT-based techniques," in *Proceedings of the International Conference on Computer-Aided Design*, ACM, 2012, pp. 436–442.

[91] H. Lin, P. Li, and C. Myers, "Verification of digitally-intensive analog circuits via kernel ridge regression and hybrid reachability analysis," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, May 2013, pp. 1–6.

[92] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, May 1, 1992.

[93] Mentor Graphics. (2016). Prologue: The 2016 Wilson Research Group Functional Verification Study, (visited on 12/10/2017).

[94] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 1998, 322 pp., ISBN: 978-0-262-19398-6.

[95] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, Feb. 2015.

[96] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis,

"Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," Dec. 5, 2017. arXiv: 1712.01815 [cs].

[97] Dhariwal, Prafulla, Hesse, Christopher, Klimov, Oleg, Nichol, Alex, Plappert, Matthias, Radford, Alec, Schulman, John, Sidor, Szymon, and Wu, Yuhuai. (2017). OpenAI Baselines, GitHub, (visited on 04/20/2018).

[98] A. Hill, A. Raffin, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable Baselines," *GitHub repository*, 2018.

[99] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the Surprising Behavior of Distance Metrics in High Dimensional Space," in *Database Theory — ICDT 2001*, J. Van den Bussche and V. Vianu, Eds., red. by G. Goos, J. Hartmanis, and J. van Leeuwen, vol. 1973, Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 420–434, ISBN: 978-3-540-44503-6.

[100] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," Dec. 19, 2013. arXiv: 1312.5602 [cs].

[101] B. Muldrey. (). Circuitgym, (visited on 10/28/2018).

[102] ——, (). Pyspectre, (visited on 10/28/2018).

[103] (). Libpsf, (visited on 10/28/2018).

[104] ——, "Mixed Signal Design Validation Using Reinforcement Learning Guided Stimulus Generation for Behavior Discovery," in *Proceedings IEEE VLSI Test Symposium*, Monterey, CA, USA: IEEE, Apr. 25, 2019.

[105] K. J. Åström, "Optimal control of Markov processes with incomplete state information," *Journal of Mathematical Analysis and Applications*, vol. 10, no. 1, pp. 174–205, Feb. 1, 1965.

[106] H. Johansson. (Mar. 11, 2019). PSF simulation data c++ library. Contribute to henjo/libpsf development by creating an account on GitHub, (visited on 03/12/2019).

[107] (). Boost C++ Libraries, (visited on 03/12/2019).

[108] (). NumPy — NumPy, (visited on 03/12/2019).

[109] M. Plappert. (2016). Keras-rl.

[110] B. Muldrey. (). Waverunner 0.2b4, (visited on 03/08/2019).

VITA

Barry attended Jesuit High School where he received a college preparatory education, including five years studying the classics, both latin and greek. There Barry performed music often, a banjo-based and cafeteria-centric adaptation of *The Devil Went Down to Georgia* probably can be credited with his election for student council vice-president. In 2003, Barry accepted a Presidential Scholarship to attend the University of Southern California. Almost certainly granted for his academic achievements and not for his musical talents, he was nevertheless determined to study Jazz and Studio Guitar. He returned to New Orleans after a year at USC to accept an opportunity to work under Mark Bingham and John Fischbach at Piety Street Recording studios. There, a closet of broken electronics piqued his interest, and he began tinkering. The fall of hurricane Katrina in 2005 threw the city into disarray and prompted Barrys reentry to academics when he enrolled in the University of New Orleans electrical engineering program. There, he was fortunate to conduct research in neural networks and participate in team robotics. He received his B.S. degree under the supervision of Dr. Edit Bourgeois and Dr. Dimitrios Charalampidis. After attempting a startup in recording studio technology, Barry applied to the masters pro- gram at Georgia Tech, and soon after arrival, passed the preliminary qualifying exam for the Ph.D. Supervised by Dr. Abhijit Chatterjee, he conducted research in circuit model validation, debugging, and machine learning for testing analog and mixed-signal systems. While spending the summer of 2013 in Santa Clara, CA conducting research for Intel Corporation, Barry was very lucky to meet his wife, Ashley King, in neighboring Oakland where she was beginning her own career as a psychiatric nurse-practitioner.